JOHANN WOLFGANG GOETHE UNIVERSITY OF FRANKFURT

MASTER THESIS

Single-task and multi-task transfer learning in a multi-source context

Author:
Daniel PIETSCHMANN &
Yannic VORPAHL

Supervisor:
Prof. Dr. Gemma ROIG

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Chair for Data Analytics and Modeling
Computer Science Department

December 21, 2021

JOHANN WOLFGANG GOETHE UNIVERSITY OF FRANKFURT

Abstract

Faculty of Computer Science and Mathematics
Computer Science Department

Master of Science

Single-task and multi-task transfer learning in a multi-source context

by Daniel PIETSCHMANN & Yannic VORPAHL

When performing transfer learning in Computer Vision, normally a pretrained model (source model) that is trained on a specific task and a large dataset like ImageNet is used. The learned representation of that source model is then used to perform a transfer to a target task. Performing transfer learning in this way had a great impact on Computer Vision, because it worked seamlessly, especially on tasks that are related to each other. Current research topics have investigated the relationship between different tasks and their impact on transfer learning by developing similarity methods. These similarity methods have in common, to do transfer learning without actually doing transfer learning in the first place but rather by predicting transfer learning rankings so that the best possible source model can be selected from a range of different source models. However, these methods have focused only on single-source transfers and have not paid attention to multi-source transfers. Multi-source transfers promise even better results than single-source transfers as they combine information from multiple source tasks, all of which are useful to the target task. We fill this gap and propose a many-to-one task similarity method called MOTS that predicts both, single-source transfers and multi-source transfers to a specific target task. We do that by using linear regression and the source representations of the source models to predict the target representation. We show that we achieve at least results on par with related state-of-the-art methods when only focusing on single-source transfers using the Pascal VOC and Taskonomy benchmark. We show that we even outperform all of them when using single and multi-source transfers together (0.9 vs. 0.8) on the Taskonomy benchmark. We additionally investigate the performance of MOTS in conjunction with a multi-task learning architecture. The task-decoder heads of a multi-task learning architecture are used in different variations to do multi-source transfers since it promises efficiency over multiple single-task architectures and incurs less computational cost. Results show that our proposed method accurately predicts transfer learning rankings on the NYUD dataset and even shows the best transfer learning results always being achieved when using more than one source task. Additionally, it is further examined that even just using one task-decoder head from the multi-task learning architecture promises better transfer learning results, than using a single-task architecture for the same task, which is due to the shared information from different tasks in the multi-task learning architecture in previous layers. Since the MOTS rankings for selecting the MTI-Net task-decoder head with the highest transfer learning performance were very accurate for the NYUD but not satisfying for the Pascal VOC dataset, further experiments need to varify the generalizability of MOTS rankings for the selection of the optimal task-decoder head from a multi-task architecture.

Acknowledgements

We would like to thank Prof. Dr. Gemma Roig for her support. With her scientific guidance she helped us a lot in our research project and had a great impact on this work. Our sincere thanks go to Kshitij Dwivedi, who constantly supported us with feedback, advise and time for discussions on sometimes even a biweekly basis. Furthermore, we would like to thank the team of Aisel-Infra for providing us with the g4-server and the g5-server. Without this powerful hardware the experiments in this thesis would not have been possible. We furthermore would like to thank all the unnamed people who supported us in the time of writing our thesis – be it with encouragement, a scientific discussion or an advice about our writing.

Contents

List of Figures

List of Tables

Acronyms

R^2	R-squared. 2, 21, 41
AHP	analytical hierarchy process. 24
ANN	artifical neural network. 5, 9
BF	Bayes factor. 2, 20, 21, 40, 41, 48, 49
BIP	binary integer programming. 25
CNN	convolutional neural network. 5, 6, 13
CPU	central processing unit. 36
CV	Computer Vision. 1–6, 9, 19, 20, 23, 26, 55
DD	duality diagram. 28
DDS	duality diagram similarity. ix, 3, 27–29, 36, 37, 40, 49, 74–76, 78
DL	Deep Learning. 4–6, 36
DNN	deep neural network. 28–30
dp	depth. 53, 57, 61, 66
fe	feature extraction. xiii, 65–69, 71, 76
ft	fine tuning. xiii, 65–71
GPU	graphics processing unit. 25, 36, 65, 66
hp	human parts. 53, 54, 59, 62, 67–69, 76
HRNet	High resolution network. ix, 19, 20, 55, 56, 77
IoU	Intersection over Union. 54, 55
mIoU	mean Intersection over Union. xii, xiii, 54, 55, 59, 66–71
ML	machine learning. 5–7, 9
MOTS	Many-To-One Task Similarity. viii, ix, xi–xiii, 2, 3, 31–38, 40–53, 55, 60, 68–71, 73–78
MTI-Net	Multi-Scale Task Interaction Network. xii, 2, 14, 17, 52, 53, 55–61, 65, 68, 69, 71–73, 76, 77
MTL	Multi-Task Learning. ix, 2–4, 6, 8–11, 13, 14, 17, 27, 35, 36, 55, 69, 73, 76, 77
NLP	natural language processing. 9
NYUD	NYU-depth. xii, 3, 35, 36, 53, 56, 58–61, 65, 68, 69, 76

PAD-Net	Pattern and Distillation Network. 13, 17, 77
PAP-Net	Pattern-Affinitive Propagation Network. 14, 77
Pascal VOC	Pascal Visual Object Class. xi–xiii, 3, 34–37, 40, 42, 44, 49, 53, 55–61, 65, 68–71, 74–76, 78
RAM	random-access memory. 65
RDM	Representation Dissimilarity Matrix. xi, xii, 26, 32, 33, 37–53, 59, 60, 71, 74–76, 78
ReLU	rectified linear unit. 60
ResNet	Residual neural network. 77
RGB	red-green-blue. 5
RL	reinforcement learning. 9
RMSE	root mean squared error. xiii, 53–55, 66–70
RSA	Representation Similarity Analysis. 3, 19, 25–27, 29, 75
sl	saliency. 53, 54, 57, 59, 62, 67, 71, 76
sn	surface normals. 53, 57, 59, 61, 62, 66–69, 71, 76
ss	semantic segmentation. 53, 69, 71
ST	single-task learning. 2, 10, 11, 17, 55, 56, 73
TL	transfer learning. 1–4, 6–9, 19, 23, 26–29, 34, 36, 37, 39, 42, 46, 52, 53, 56, 57, 61, 65, 68, 69, 71, 73, 76–78
TLP	transfer learning performance. xi, xii, 1–3, 19, 23, 26–28, 31–34, 37, 39, 40, 42–44, 46, 48, 50, 51, 59, 61, 68, 69, 71, 73, 74, 76–78
TN	transfer net. 25, 52, 56, 62–68
TRL	task-relationship learning. 11, 17

Chapter 1

Introduction (Yannic)

Computer Vision (CV) has undergone a rapid development in recent years, becoming an important aspect in the development of artificial intelligence, for example in autonomous driving and the creation of deepfakes. Nowadays, new innovative approaches in CV are introduced on an annual basis, with the goal to outperform previous state-of-the-art methodologies. One such technology is transfer learning (TL) that has had a powerful impact in improving the CV model using representations which were learned by another model [17][32][37]. TL is especially important, when data quality is not good enough, when not enough data is available to train a model or when computational resources are limited. TL tackles these issues through the support of an additional model. Generally, when TL is performed, there is a model, that needs to be trained for a specific task called the target task, and a supporting model is used that has already been trained. This pretrained model is called the source model, and the task on which it is trained is the source task. A source model is usually trained on a large dataset like ImageNet [14]. The learned representations of the source model are then transferred to a target model and then further processed from the target model for the prediction of the target task. This current state-of-the art method has been proven to be applicable for a range of different tasks [32][37][65] and using this approach promises several advantages. Computational resources are saved; use-cases with smaller datasets are possible; the overall training time is reduced, and the result of the target model is essentially improved through avoiding model training from scratch.

TL has led to great results and is currently used in practice as a quasi-standard. Unusual to other CV technologies, this approach has not changed over the last years. One of the reasons for the functionality of TL is assumed to be linked to the relationship between different CV tasks with the assumption, that related CV tasks promise better transfer results, than unrelated tasks that are linked to negative transfer [84][17][40]. Negative transfer occurs, when a respective transfer from one source-model makes it more difficult to solve a problem for the target model [48]. The question of how to choose the most suitable source model and how to minimize the risk of negative transfer constitutes an ongoing discussion [84][17][40]. A measurement to quantify the relationship between different tasks is thus a potential solution and currently an ongoing topic [17][40][60][84]. Taskonomy [84] is one of the first experiments, that analyzed the TL impact of different types of source-tasks (26 tasks in total) in different combinations to one specific target task by measuring the TLP. By doing that, Zamir et al. [84] Taskonomy represented the best performing source-tasks linked to specific target tasks and the interrelations between different CV tasks.

Taskonomy has the great potential to be used as an orientation to select the most suitable source-tasks, that promise the best TLP to one target task. Despite these findings, it only works for a new target task by doing all the transfers from scratch again and thus obligates all source task combinations to be trained again, which is cumbersome and time-consuming. Alternative approaches have been produced to overcome this obstacle by using the similarity of the tasks used to do TL without doing the actual transfer in the first place [17][40]. The similarity approach is driven by the assumption, that more similar (related) CV tasks promise better TL results [84]. Current similarity methods provide accurate TL predictions quickly [17][40][60]. Nevertheless, each similarity method only includes single-source transfer predictions without including the multi-source transfers represented in Taskonomy [84].

Multi-source transfers contain the potential to improve the performance of the target model even further, because representations of different source tasks may contain useful information that leads to a better transfer to the target task. We therefore propose a new similarity method to fill this gap: MOTS.
MOTS estimates the similarity score between source and target tasks by applying a linear fit on the representation of the sources to predict the target representation. The linear fit is then evaluated by the R-squared (R^2) method and the Bayes factor (BF). In this way, we obtain rankings of different source-task combinations used for the prediction of a specific target task and consequently obtain the best sources for multi-source transfers on one specific target task.

Choosing the right setting for the implementation of multi-source transfers is especially complex. Performing multi-source transfers pretrained on individual single-task learning (ST) architectures leads to higher computational costs and longer training times due to the linear increase of model parameters to train with the number of source tasks. MTL is an alternative approach that is still relatively new in CV [10][52][74]. A MTL-architecture usually consists of an encoder-decoder structure in which an encoder is used for a shared representation of multiple tasks, and a decoder is then used to obtain several task-specific decoder heads [73]. A MTL-architecture provides several advantages over a ST architecture: It handles several tasks in one architecture; similar representations from each ST architecture are only trained once, and it provides multiple target-task outputs through its decoder heads [73]. A MTL-setting thus enables through its shared representation a higher training speed and lower computational costs, and it avoids a linear increase in model parameters by the number of source tasks. A MTI-Net is currently a state-of-the-art MTL architecture, and it outperforms similar counterparts and ST architectures [73]. We use MOTS with the MTI-Net as our architecture to validate our method. We include one target task and the task-specific decoder heads of the MTI-Net in different combinations as our source-tasks to estimate the similarity score between source and target. By doing this, we obtain a ranking showing which combinations of decoders are most suitable for multi-source transfer to a target. MOTS is then evaluated by comparing its rankings with the rankings that are obtained using the TLP from Taskonomy [84].

We suspect that our method suggests better source models for TL than related methods by including multi-source transfers. We also expect better results using multi-source transfers rather than only single-source transfers.

To prove our claim we will answer the following research question in this thesis:

Does our similarity method promise state-of-the-art results in TLP prediction when using a linear fit between source and target task for single- as well as multi-source tasks?

The research question can be broken down into the following areas, each of which needs to be answered individually:

1. How well does MOTS perform in single-source TLP prediction?
2. How well does MOTS perform in multi-source TLP prediction?
3. How well does MOTS perform in TLP prediction using the task-decoder heads of a mulit-task architecture as sources?

To address these matters, we proceed as follows. We compare the single-source TLP of MOTS with other state-of-the-art methods. We then apply MOTS to multi-source transfers and compare its results with state-of-the-art single-source transfer methods. Next, we validate the performance of MOTS in a practical, validation environment by using a new architecture trained on two datasets (NYUD and Pascal VOC). The task-specific representations obtained from the model are then used in our MOTS method to create prediction rankings of different source-tasks combinations to one target task. The rankings are then compared to the actual TLP, which is achieved through the actual training of the model, and the transfers to a target task.

We first introduce the theoretical nature of our work in Chapter 2 and present the area of CV in Section 2.1, TL in Section 2.2, MTL in Section 2.3, task-relationships in Section 2.4 and the mathematical basics in Section 2.6. Following this, we showcase related works, that has already been accomplished. We introduce the ground truth of our method in Section 3.1, the Representation Similarity Analysis (RSA) method in Section 3.2, the duality diagram similarity (DDS) method in Section 3.3 and then attribution maps in Section 3.4. We then present our method MOTS in Chapter 4 and the results of all of our experiments in Chapter 5. We discuss and conclude our thesis with proposals for future work and an outlook in Chapter 6.

Chapter 2

Theoretical background

In this chapter, we provide a theoretical overview of the concepts used in our thesis. We provide information about general concepts regarding CV and Deep Learning (DL), actual literature, state-of-the-art methods in TL and mathematical concepts that we use in our analysis.
We start with the broader, more general topics of CV and TL and later dive deeper into different architectures of MTL.

2.1 Computer Vision (Yannic)

CV deals with digital images or a sequence of digital images. The main goal of CV models is hereby the understanding and the interpretation of images and the information it can extract from them[42][80]. The outcome can vary and depends on the visual task, a computer is to perform[80].

CV often refers to machines, that have the the ability to see like a human[42][80]. In defining its features, a CV system should contain the definition of "seeing" or "vision" to use as a guide. Learned-Miller [42] states that human vision does not consist of just one component, but rather exists through a set of different components, such as memory, retrieval, reasoning, estimation, recognition and coordination with other senses. He further explains that for a system to be considered to have a degree of vision, it is not mandatory to include all these components.
Huang [31] separates CV between biological science point of view and an engineering points of view. He mentions, that from a biological perspective, computational models are built to be similar or identical to the human visual system, whereas from an engineering perspective, autonomous systems are built to execute visual tasks, that a human visual perspective can perform. While the first viewpoint focuses on replicating the human visual system in an identical manner, the second viewpoint rather aims to achieve a similar or identical output between a visual machine and a human, regardless of the system design.
Fermüller and Aloimonos [22] connect vision with the two components of perception and action and link vision among others with physiology, the brain and the behaviour. They define a CV system as a system, that “interacts with the space-time in which the system exists and the establishing of relations between these representations, and the system actions” [22, p. 742]. Their focus mainly lies in the reconstruction of a behavioural framework.

CV was originally invented to transform 2D photographs back into a 3D form [41, p. 8, 9]. This is an inverse problem in which an original state is restored with insufficient information about its original form [64, p. 3]. Learned-Miller [42] calls this inference, which is assumed from a picture about the world. In solving visual tasks, CV is mentioned along with related fields like digital image processing, pattern recognition, machine learning (ML) and Computer graphics, and it uses concepts, ideas and techniques from these systems [80]. Image processing and CV are often handled in the same manner. Wiley and Lucas [80] define image processing as computational transformations performed on images, while CV is more focused on the creation of models and data extraction it obtains from images. CV and computer graphics are opposites. While CV processes images from the world in order to convert them to abstract representations, computer graphics takes an abstract representations of the world into a computer in order to turn them into images. Pattern recognition is defined as "a branch of CV, which focuses on the process of object identification through image transformation to get better image quality and image interpretation". [80, p. 30]

One of the challenges in CV involves the inverse transformation of an images back to their 3D form, which is easily manageable for the human eye but not so much for a CV model [41]. From a human perspective, the interpretation of spatial data is processed in seconds, while a CV system has limited capacities and relies on the "sensitivity of the parameters, the strength of the algorithm and the accuracy of the result" [80, p. 29]. Machines do not recognize pictures the way humans do; they represent digital images in the form of arrays with a variety of different numerical values [41, p. 2][13, p. 1], as can be seen in Figure 1. An array is described as a much larger and more complex form of an image [13, p. 1]. An image is thus defined as a representation, respectively as a picture that is visualized as a 2D projection from a 3D scene and captured by a sensor from an imaging device and consists of pixel values. Pixels are thereby discrete and responsible for the brightness of a scene. They lie in the range of $k = 2^b$ with b being the number of bits. The number of pixel values represents the resolution of an image, whereas the number of channels describe the depth of an image. A gray-scale image uses one channel, whereas a colorful image like one in red-green-blue (RGB) represents multiple channels. A gray-scale image represents the luminance at each point of a scene. A color image contains chrominance (color information) in addition to the luminance; it is therefore larger and requires more resource-intensive to be processing compared to its gray-scale counterpart [13, p. 1, 2].

Nowadays, two approaches are used to perform computational visual tasks. One of them is referred to as a classical ML approach with a set of different methods for different use cases, while the other is DL [45]. The difference between both is visualized in Figure 2. Both approaches identify features in an image, thus supporting a system through its process to its output. Mahony et al. [45] defines features as something small, visible in a particular area of an image, about which a certain statement can be made. For a classical approach, a CV expert is needed, to identify important features in an image. A classical approach is often referred to as a manual, handcrafted feature-extraction process, while no expert is needed in DL and the feature extraction process is done completely automatically. DL is therefore defined as a subset of ML and linked to artifical neural network (ANN) [45]. Especially in CV, a convolutional neural network (CNN) is often mentioned as an architecture that processes inputs using a variety of different methods in order to produce an output [45]. The basic architecture of a CNN is displayed in Figure 3. Compared to a classical ML approach, CNNs achieve better results in image classification, segmentation and in object detection, but they need data and computational power in order to be trained accurately. In this thesis, we use architectures based CNNs, and we therefore forego on a detailed description of the other approaches. Some classical methods, as well as further DL architectures are displayed in Table 2 and can be further examined in the respective papers. The following sections deal with CNN and especially with a variety of different CNN architectures being discussed currently, especially in the domain of TL and MTL.

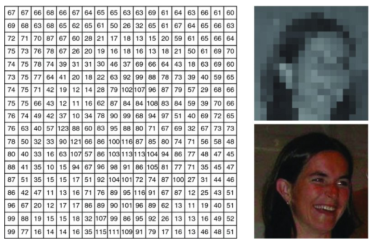


FIGURE 1: Different versions of images[13].

Classical ML	Explanation
SIFT[35]	Auto-Encoder[2]
SURF[3]	CNN[43][39]
FAST[51]	RBM[47]
Hough transforms[27]	LSTM[30][54]
Geometric hashing	RNN[54]

TABLE 2: Classical machine-learning algorithm vs. deep-learning architectures.

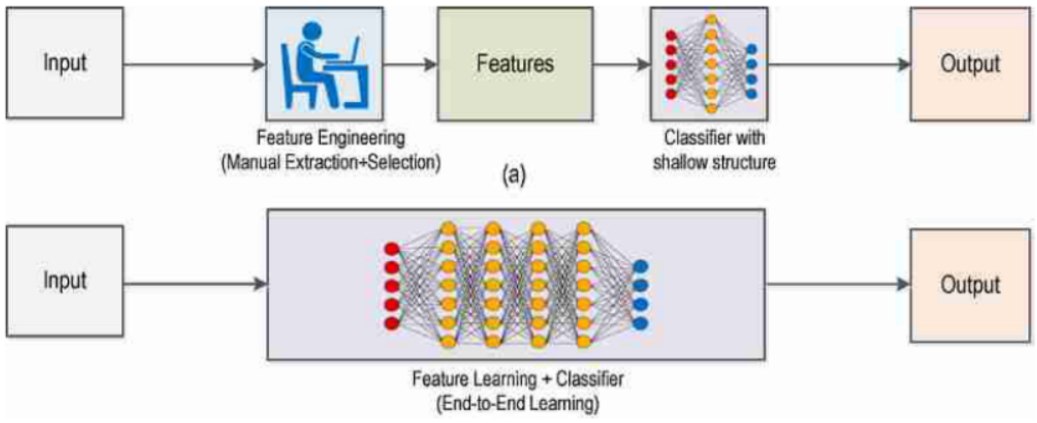


FIGURE 2: Traditional approach vs. deep learning [78].

2.2 Transfer Learning (Daniel)

In this section, we explain the theoretical overview of TL and its application areas. TL is a method used in ML that allows the use of learned knowledge from one model to be transferred to another. It is widely used in the DL and CV community as many

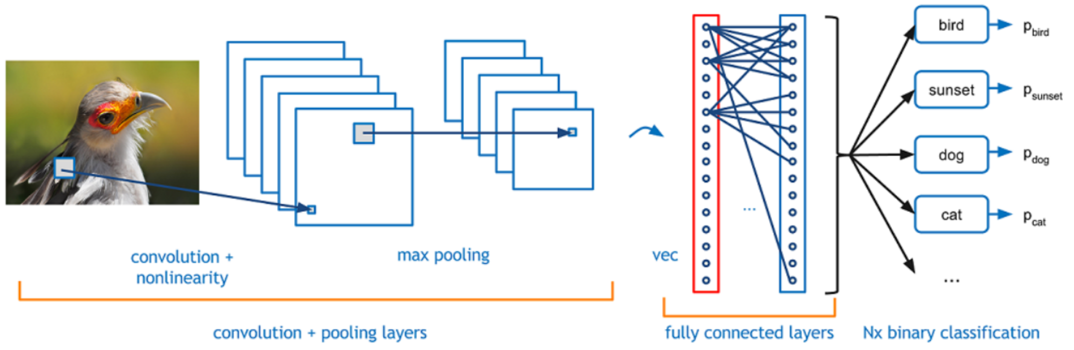


FIGURE 3: Basic CNN architecture [15] .

image-processing tasks are difficult to learn and train, so they require high computing power, which is what makes TL so attractive. Furthermore, TL is commonly used in data mining and ML applications [48].
A typical use case for TL is when there is not enough data, or the training of the model for the desired task of interest is not feasible. In cases like this, it is recommended or even necessary to use TL to acquire a model for the task of interest.
Compared to a classical ML problem, TL makes the assumption that the data used for training and the data used during inference do not need to be in the same feature space[1] nor do they need to have the same distribution. Pan and Yang [48] give a specific definition of TL as follows:

> Given a source domain D_S and learning task T_S , a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$. [48, p. 1347]

with $D_S = \{(x_{S_1}, y_{S_1}), ..., (x_{S_{n_S}}, y_{S_{n_S}})\}$, where $x_{S_i} \in X_S$ is the data instance and $y_{S_i} \in Y_S$ is the corresponding class label (assuming that the task is a classification task). Furthermore, Pan and Yang [48] define the goal of TL as being to "extract the knowledge from one or more source tasks and apply the knowledge to a target task" [48, p. 1346]. A key characteristic of TL is that it allows domains, tasks and distributions to differ between the training and testing of the model. [48] To understand the concept of TL, it is necessary to understand the concepts of positive and negative transfer: "Negative transfer happens when the source domain data and task contribute to the reduced performance of learning in the target domain" [48, p. 1354]. Positive transfer is the opposite and occurs when the source domain knowledge (data and task) improves the performance of the target task. In general, TL is applied to perform a positive transfer of knowledge from a source to a target model. In the case when a relationship between the feature spaces of two domains exists, Pan and Yang [48] call these two domains related.
The authors divide the research issues of TL into three groups based on different questions: What should be transferred?; How should this transfer be done?; and When and in which situations should the transfer occur [48]?
As shown in Figure 4, Pan and Yang [48] differentiate between three types of TL settings:

- Inductive TL

[1]The feature space is a n-dimensional space. The number of dimensions of this space is equal to the number of variables/properties [58]

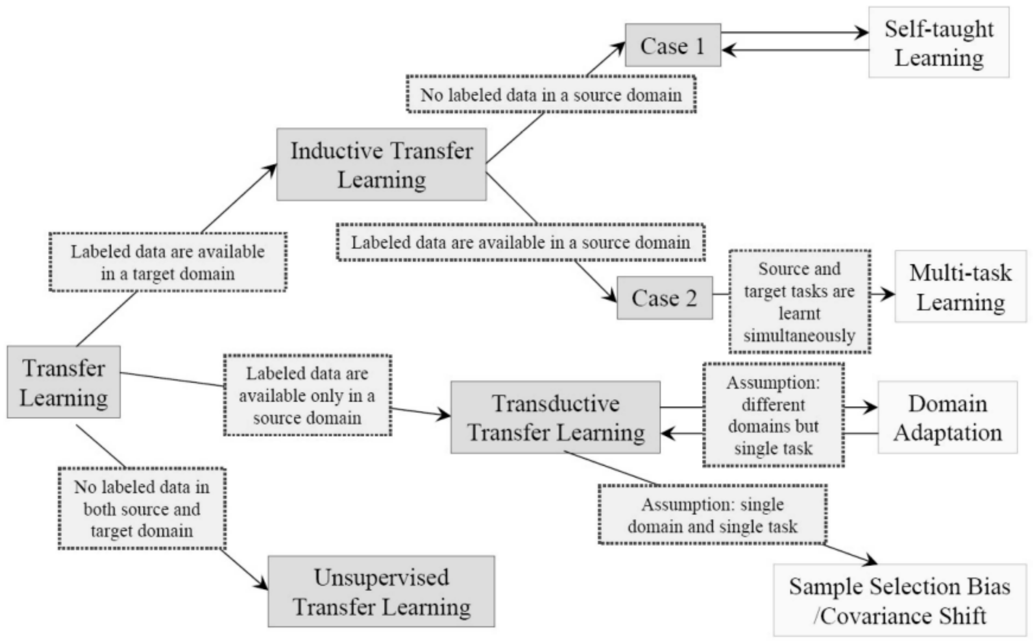

Case 1
Self-taught Learning
No labeled data in a source domain
Inductive Transfer Learning
Labeled data are available in a target domain
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Transfer Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
No labeled data in both source and target domain
Assumption: single domain and single task
Unsupervised Transfer Learning
Sample Selection Bias /Covariance Shift

FIGURE 4: An overview of different settings of transfer learning [48].

- Transductive TL
- Unsupervised TL

In **inductive TL**, target and source tasks are different, and labeled data is available in the target domain. Pan and Yang [48] further categorize inductive TL into MTL, in which case-labeled data is available in the source domain and self-taught learning – in which there is no labeled data – in the source domain.
In comparison to inductive TL, in **Transductive TL** the source and target tasks are the same, but the domains are different. Transductive TL is categorized into cases with different feature space ($X_S \neq X_T$) and cases with the same feature space using a different probability distribution of the input data ($P(X_S) \neq P(X_T)$); this is called domain adaption [48].
Unsupervised TL is similar to inductive TL in the sense that the source and target tasks are different. Unsupervised TL differs from inductive TL by using clustering and dimensionality reduction tasks instead of classification and regression tasks.
The experiments in this thesis focus only on MTL as a part of inductive TL. Therefore, the following theoretical descriptions also concentrate on just this case and do not go into detail about the other types of TL. Pan and Yang [48] define inductive TL as follows: In the source domain D_S, a source task T_S is learned. However, the goal is to train the target task T_T in the target domain D_T. This is accomplished by using the learned representations from T_S to improve the target prediction function $f_T(\cdot)$, with the assumption that $T_S \neq T_T$ is taken.
In MTL as a subcase of inductive TL it is further assumed that there is labeled training data available in D_S. Regarding the question, What should be transferred?, TL can be disassembled into four approaches:

- Instance transfer
- Feature representation-transfer

- Parameter transfer
- Relational knowledge transfer

Instance transfer involves the transfer of labeled data from the source to the target domain. Feature representation transfer transfers the learned features from the source domain to the target domain to improve the prediction in the target domain. Parameter transfer reuses learned and shared parameters in the target domain, and relational-knowledge transfer establishes relations between source and target domain. Of these categories, this thesis only focuses on feature representation transfer in supervised learning, which works by extracting a low-dimensional representation from T_S. In an optimal case, this low-dimensional representation is shared across different but related tasks so that the knowledge from it can be transferred to T_T by performing positive transfer instead.
TL has been widely applied in different research areas, for example in text and natural language problems such as domain adaption for sentiment classification [6], document classification [11] and CV tasks (self-taught clustering of images [12] and image clustering [82]).

2.3 Multi-Task Learning (Yannic)

TL and MTL are closely related. For our methodology, the selection of the right MTL architecture plays an important role, which is why a basic understanding of MTL is necessary. Before diving into the topic, MTL is discussed in many different areas of ANN (natural language processing (NLP), reinforcement learning (RL), CV)[10]. This Section deals with the topic with regards to the field of CV

Ruder [52] describes MTL as a framework that takes multiple tasks into one shared representation to predict multiple target tasks, which consists of more than one loss function that needs to be optimized. Crawshaw [10] highlights the original idea behind MTL to imitate the human way of learning. A human brain combines and uses previously acquired knowledge of various tasks to perform multiple tasks at once by using the same layers for different tasks in earlier regions of the brain [74].
Similar to Ruder [52], he defines MTL as a sub-field of ML in which multiple tasks are performed simultaneously by a shared model. Ruder [52] describes MTL among other things from a ML perspective "as a form of inductive transfer"[52, p. 2]. Inductive transfer is described as a form, that improves a model through inductive bias. Inductive bias is further described as a measure, that favors one hypothesis over the other.

The challenge in developing a MTL design in CV consists of two things:

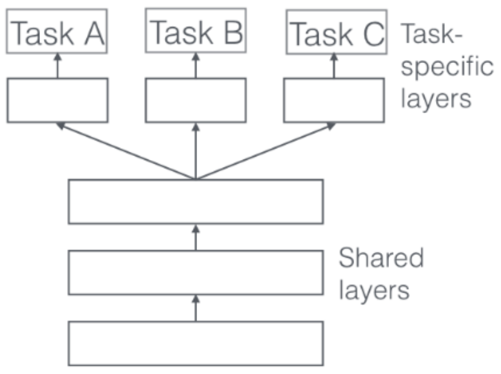
- Selection of the appropriate design structure of the architecture.
- Implementation of optimization techniques, that need to be performed in parallel on different tasks without being disadvantageous for at least one of them.

Current MTL implementations are classified as follows [10]:

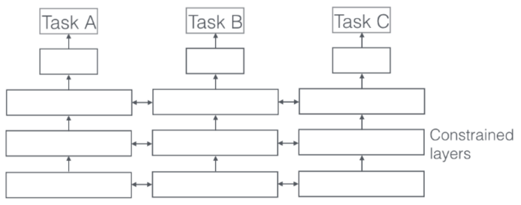
- Shared trunk: architectures with a global feature, which is shared across all tasks and an individual output branch for each task
- Cross-talk: individual branches for each task with shared information between parallel layers

- Prediction distillation: preliminary predictions for each as initiator to predict final output for certain tasks
- Task routing: parameter sharing between tasks at the feature level and not layer level
- Single tasking: single task, that is used for multiple task predictions

A more abstract representation of a MTL frameworks is displayed in Figure 5 and consists of hard parameter sharing (see Table 5a) and soft parameter sharing (see 5b. Hard parameter sharing is accomplished through two steps. First, it involves model parameters which are shared between different tasks at the beginning. Second, each task branches to its own network and generates its own task-specific output. Soft parameter sharing provides each task with its own independent network. Each independent network tends to communicate with the others in order to maintain a similarity with each other [52]. Ruder [52] underlines the issue, especially regarding architectures with hard parameter sharing, which tend to work seamlessly only for closely related tasks. He points out, that these approaches hinder the implementation of less related tasks, and he identifies task similarity, task relationships, and task hierarchy as three important points that need to be researched further, because of negative transfer [52]. Negative transfer happens when tasks are not closely related. This can be seen as a win-loss situation in which one task may increase in accuracy while the other one decreases. As a result, performance degradation occurs with some tasks, while others increase in accuracy. This makes the usage of a MTL approach for some tasks unnecessary when a ST approach shows better performance [10]. Vandenhende et al. [74] therefore argue that suitable task-groupings need to be selected in advance to prevent negative transfer and they state as well that task relatedness and task similarity need to be considered for optimal task groupings, but they also emphasize suitable layers in which the sharing should occur. According to the authors, fewer related tasks can be implemented in that way by having their own layers. Vandenhende et al. [75] proposes a new MTL-framework that is considered to be an encoder-decoder-based approach for dense prediction tasks. The structure is visualized in Figure 6.



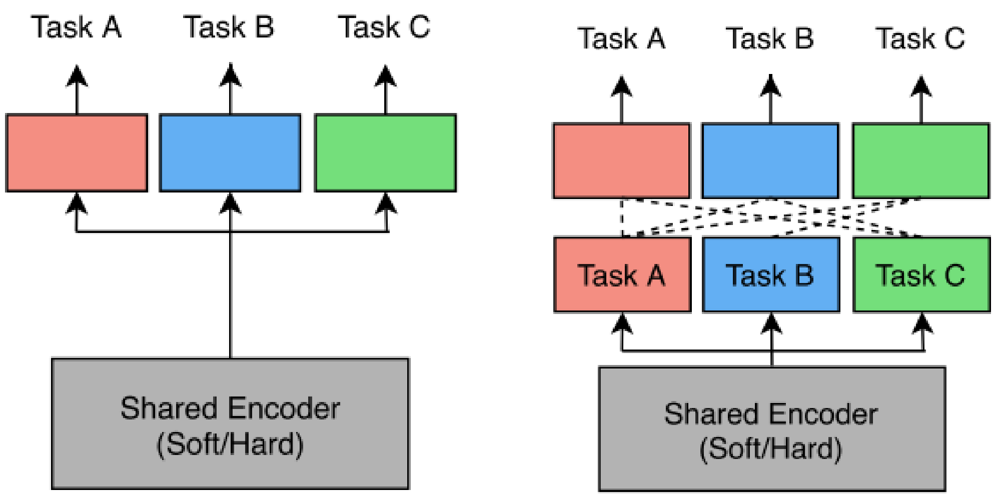
(a) Hard parameter sharing in MTL



(b) Soft parameter sharing in MTL

FIGURE 5: MTL approaches in deep neural networks [52].

Having a MTL architecture for related tasks, promises several advantages. A shared



(a) Encoder-focused model (b) Decoder-focused model

FIGURE 6: Encoder- and decoder-focused MTL-models [75].

representation between them lead according to Ruder [52] to a model that generalizes better compared to a ST model. This can for instance reduce the risk of overfitting [52].
Crawshaw [10] and Vandenhende et al. [74] describe MTL as a promising method for increasing learning speed for related tasks but also for downstream tasks. MTL is capable of drastically decreasing the amount of computational time since training of models for each task is no longer mandatory.
Currently, advancements in MTL are being made in the areas of architecture, optimization and task-relationship learning (TRL) [10]. Crawshaw [10] connects hard parameter sharing to the area of MTL architectures and soft parameter sharing to the area of MTL optimizations. The author [10] describes both areas and TRL; this is further described in Section 2.4, as the guiding framework in MTL, and contradict the usual approach that separates a MTL structure into hard and soft parameter sharing.

2.3.1 Cross-Stitch-Network (Daniel)

The cross-stitch network is an encoder-focused MTL architecture that uses soft-parameter sharing. It combines multiple networks by using so-called cross-stitch units and is end-to-end trainable. Cross-stitch networks can be used to estimate the optimal combination between shared and task-specific representations in a MTL setup. The cross-stitch unit uses a linear combination of activations of different networks to learn a shared representation [46].
Figure 7 shows a cross-stitch unit that combines two networks, each regarding its own task.
The cross-stitch unit works as follows: Given the activation maps x_A from network A and the x_B from network B (both from layer l) the linear combinations $\tilde{x}_A$ and $\tilde{x}_B$ are calculated by modulating the activations by α (Equation 2.1):

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_A A & \alpha_A B \\ \alpha_B A & \alpha_B B \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix} \tag{2.1}$$

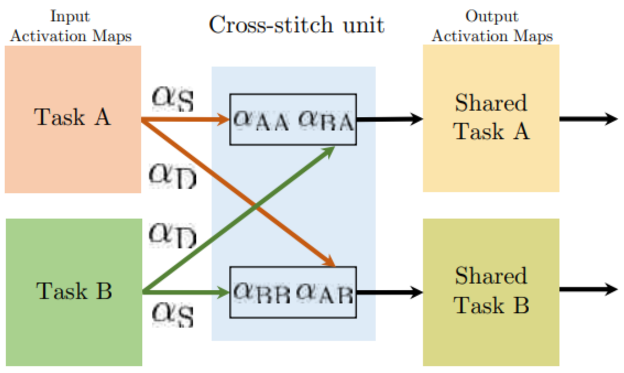


FIGURE 7: Cross-stitch unit [46].

After $\tilde{x}_A$ and $\tilde{x}_B$ are calculated, they are used as input for the next layer.

By changing α the cross-stitch unit supervises itself as to how much sharing is necessary. By selecting α, the model can choose the optimal combination between shared and task-specific parts [46].
The main disadvantage of the cross-stitch network is its scalability problem and its restriction on the usage of local information. The scalability problem is due to the fact that the network´s size increases linearly with the number of tasks. The second disadvantage is that the network can only use limited information while modulating α and therefore has only a small receptive field [73].

2.3.2 Pattern and Distillation Network (Yannic)

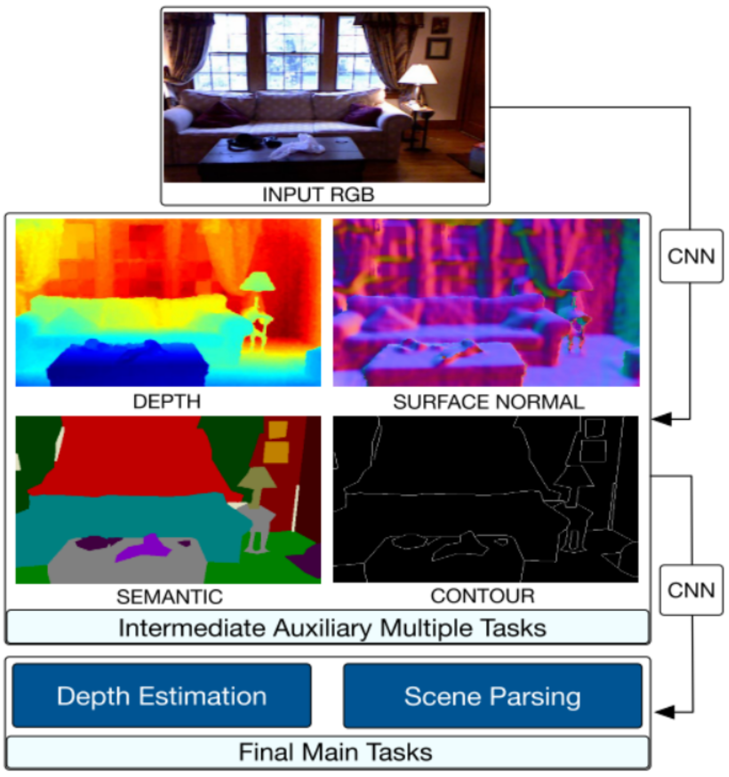


FIGURE 8: Idea of PAD-Net [17].

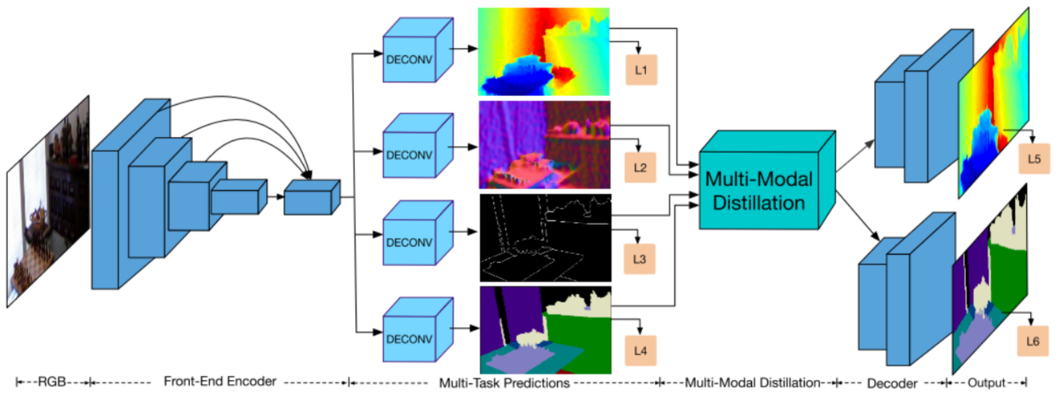


FIGURE 9: Pad-Net framework[17].

Pattern and Distillation Network (PAD-Net) is a decoder-focused MTL architecture that supports target task predictions with a distillation network. Figure 8 visualizes the idea behind PAD-Net and auxiliary tasks, in which one input image is being processed through different CNNs, that solve individual tasks. These intermediate task predictions are then used as multi-modal data input and further processed with a multi-modal distillation module in order to use the important information from each intermediate task for the prediction of the final multi-task output. Xu et al. [81] argue that the knowledge shared by different intermediate tasks creates a generalizable MTL model with the advantage of being capable to produce different target tasks.

Xu et al. [81] describe the advantages and potentials of the multi-modal approach from three perspectives:

- Multi-modal data improves the performance of deep predictions due to the complementary information they have for the final output.
- Information from related tasks that are not being predicted as final tasks, can also improve the performance further if used as intermediate tasks.
- The design of the multi-modal module facilitates the communication and sharing of information from the multi-modal data, that consists of different visual tasks.

The authors add that their approach improves optimization of the frontend network, provides valuable initial predictions and promises a better handling of task information derived from the intermediate tasks in earlier layers [81].

Figure 9 illustrates the PAD-Net architecture and its four main components. The multi-modal distillation module plays a significant role in this framework, because the final multi-task output depends on its design and the way the information from different intermediate tasks is combined for the final task predictions.
PAD-Net achieves a better result than the baseline network with any design structure shown in Figure 10. The authors also demonstrated, that a higher number of related intermediate tasks have a positive impact on the result of the network. Furthermore, the authors proved that the multi-task outputs scene parsing and depth estimation achieve better results, than state-of-the-art approaches. There is no performance degradation when both tasks are predicted in one model at the same time.

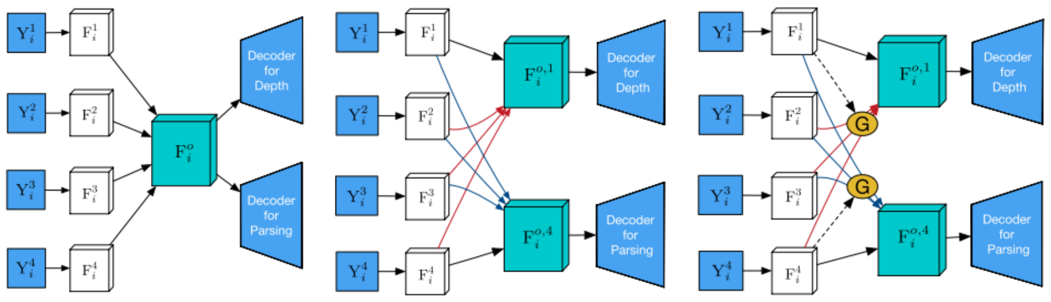


FIGURE 10: PAD-Net distillation modules [81].

2.3.3 Pattern-Affinitive Propagation Network (Daniel)

PAP-Net is a decoder-focused network for joint training of the tasks depth estimation, surface normal prediction and semantic segmentation. In this approach, the researchers utilized cross-task affinity patterns to jointly predict all three tasks. The idea of cross-task affinity patterns is to use non-local affinities between tasks to improve the joint prediction of all tasks compared to prediction network [85]. The goal is to use the affinity patterns between tasks and improve the prediction performance of each individual task based on this information.
The Pattern-Affinitive Propagation Network (PAP-Net) architecture is fully trainable in an end-to-end manner. It is visualized in Figure 11. The process used in PAP-Net contains four main steps [85]:

1. Learn affinity matrices for each task by using the affinity learning layer to store pixel pairwise similarities.
2. Combine the affinity matrices of the different tasks in an adaptive manner.
3. Use the specific diffusion layer of each task to propose the learned affinities to the features.
4. Use diffused task-specific features to produce final outputs of tasks.

The main contributions are the proposition of the PAP-Net method, the proposition of the two-stage affinity propagation and the experiments validating the PAP-Net method. Advantages of the PAP-Net method are that the network learns non-local affinities and uses these affinities for cross-task improvement. Further, it is a data-driven method that does not need task-specific supervision, which makes it easier to handle in training [85].

2.3.4 Multi-Scale Task Interaction Network (Yannic)

MTI-Net was developed to be a MTL architecture and is used at different scales. It follows the idea of current MTL approaches described in Section 2.3.2 and is considered to distill information from initial task predictions. However, what makes MTI-Net different is the consideration of task-interactions at different scales as well as is shown in 13. Vandenhende, Georgoulis, and van Gool [73] showed that pattern affinities at a certain scale do not guarantee similar behaviour at other scales, and they therefore proposed a new design that considers pattern affinities not only at the task level but also at the scale level [73]. The authors [73] explain their assumption using an illustration, which is displayed in Figure 12a:

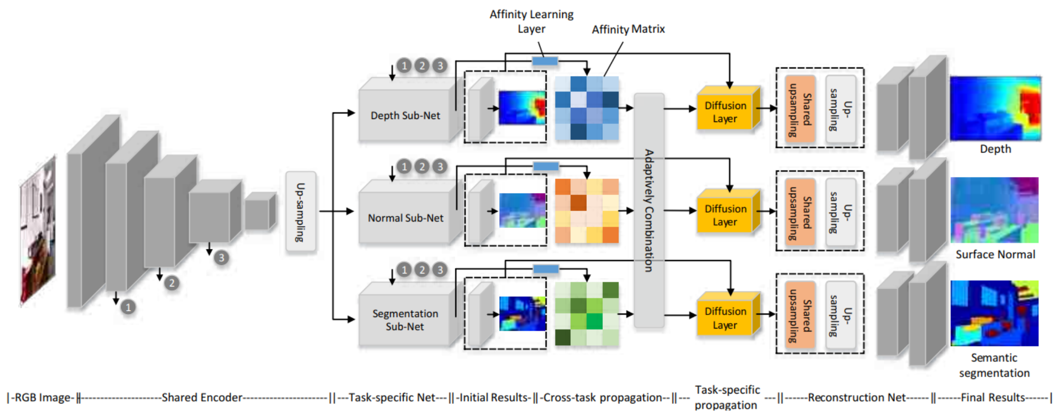


FIGURE 11: PAP-Net architecture [85].

"The local patches in the depth map provide little information about the semantics of the scene. However, when we enlarge the receptive field, the depth map reveals a person's shape, hinting at the scene's semantics." [73]

A certain size of receptive field can hide information that may be necessary for the accurate prediction of a certain visual task. Increasing that size could provide the correct result. However, task information from a certain scale that may hinder an accurate performance for a certain task, can be useful for another. The authors therefore show that visual tasks may influence one another differently depending on the receptive field size. Vandenhende, Georgoulis, and van Gool [73] validate their assumptions through an experiment, which is illustrated in Figure 12b:

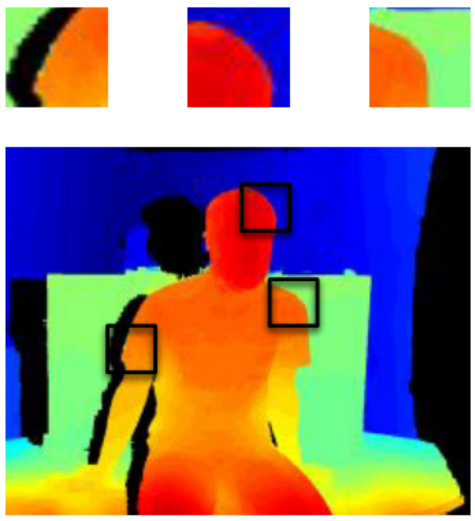
"We measure the pixel affinity in local patches on the label space of each task, using kernels of fixed size. The size of the receptive field can be selected by choosing the dilation for the kernel." [73]

The results show a high pixel affinity for each task combination for a certain dilation value, but they also demonstrate the dependency of pattern affinities with respect to a receptive field size. The experiment therefore supports the statement of Vandenhende, Georgoulis, and van Gool [73] that pixel affinities between two different tasks can vary depending on the receptive field size, which is determined by the kernel dilation [73]. Based on these findings, the authors [73] suggest a model that distills information of different tasks at multiple scales. They propose an architecture for this procedure that is built around a backbone network, a multi-modal distillation unit, a feature propagation module and a feature aggregation unit.

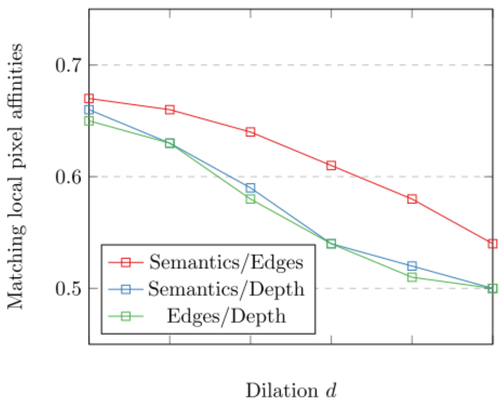
The backbone network is used to extract feature representations at multiple scales from an input image. Feature representations at each scale are then used to do initial task predictions with the output being stored in task-specific heads at different scales. [73].

A multi-modal distillation unit is thereafter used to refine feature representations that were extracted at multiple scales. The following formula is used to find the distilled task feature $F^{\circ}_{k,s}$ for task k at scale s [73]:

$$F^{\circ}_{k,s} = F^{i}_{k,s} + \sum_{l \neq k} \sigma(W_{k,l,s} F^{i}_{l,s}) \odot (W^{'}_{k,l,s} F^{i}_{l,s}) \tag{2.2}$$

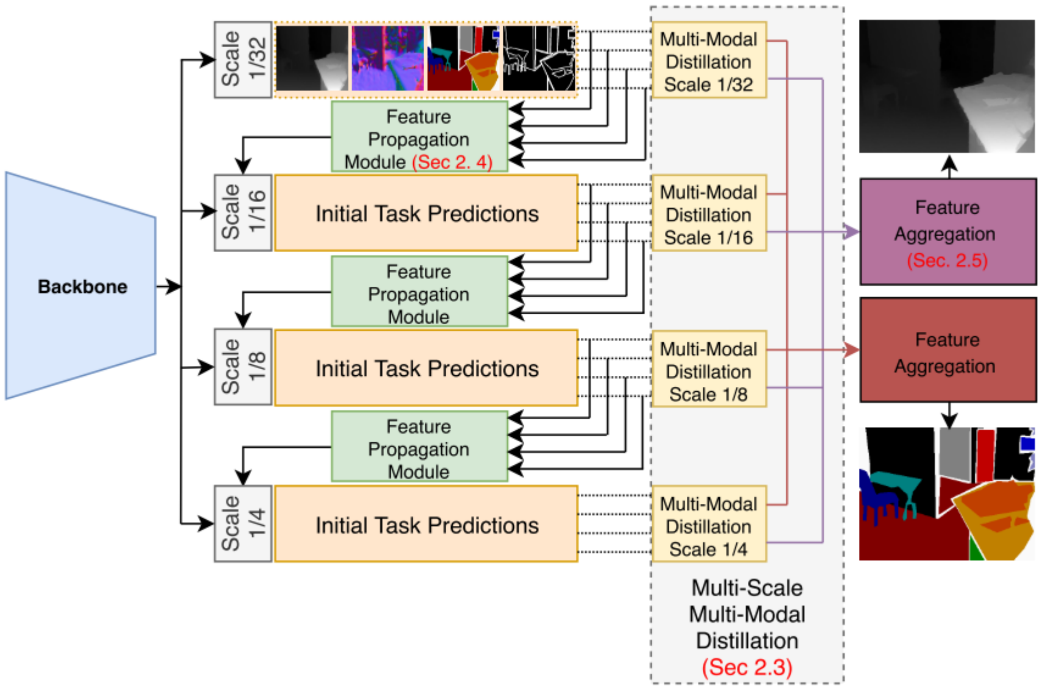


(a) Local patches from a depth map.



(b) Pixel affinities for visual tasks with different dilations.

FIGURE 12: Task-interactions at different scales [73].



Backbone
Scale 1/32
Scale 1/16
Scale 1/8
Scale 1/4
Feature Propagation Module (Sec 2. 4)
Feature Propagation Module
Feature Propagation Module
Initial Task Predictions
Initial Task Predictions
Initial Task Predictions
Multi-Modal Distillation Scale 1/32
Multi-Modal Distillation Scale 1/16
Multi-Modal Distillation Scale 1/8
Multi-Modal Distillation Scale 1/4
Multi-Scale Multi-Modal Distillation (Sec 2.3)
Feature Aggregation (Sec. 2.5)
Feature Aggregation

FIGURE 13: Visualization of MTI-Net architecture.

where $\sigma(W_{k,l,s}F^i_{l,s})$ returns a per-scale spatial attention mask that is applied to the task features $F^i_{l,s}$ from task l at scale s [73].

A feature propagation module is then used to link task features from previous

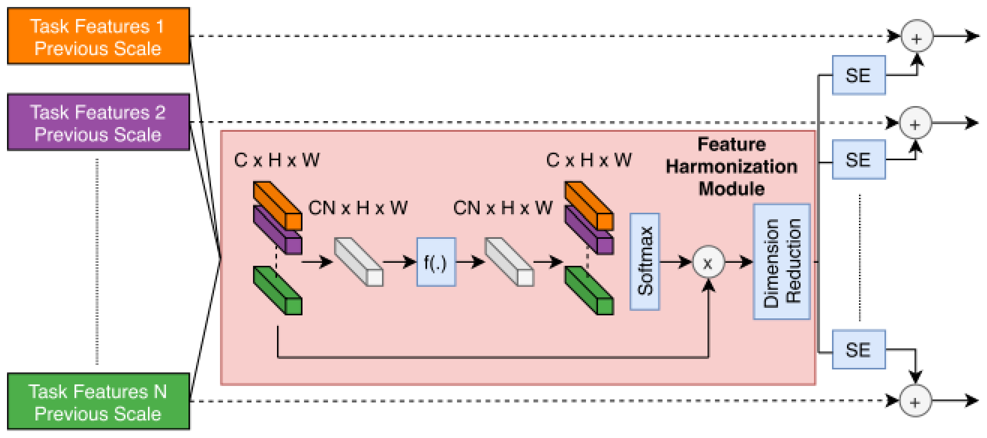

Task Features 1 Previous Scale
Task Features 2 Previous Scale
Task Features N Previous Scale
C x H x W
CN x H x W
f(.)
CN x H x W
C x H x W
Softmax
x
Feature Harmonization Module
Dimension Reduction
SE
SE
SE
+
+
+

FIGURE 14: Feature propagation M´module.

lower resolution scales that were previously refined by a multi-modal distillation unit with features from a subsequent higher resolution scale that were extracted from the backbone as described in Figure 14. According to Vandenhende, Georgoulis, and van Gool [73], this step is performed to circumvent poor initial task predictions due to the limited receptive field of view at higher resolution scales. The entire process is depicted in Figure 14 and described in more detail in [73].
The distilled task features at each scale are sampled up to the highest scale and then concatenated, resulting in a "final feature representation for every task"[73, p. 9]. Vandenhende, Georgoulis, and van Gool [73] additionally mention that in addition to initial task predictions, their model allows auxiliary tasks to be implemented, as is the case with PAD-Net.
The logical innovative design behind MTI-Net is also reflected in the model performance, showing results that are at least on par with related architectures with advantages of a smaller footprint, a reduced number of calculations and a better performance than ST architectures [73].

2.4 Task-Relationship Learning (Daniel)

In contrast to MTL architecture design and changing optimization techniques, TRL is another approach of MTL. Crawshaw [10] divides TRL into three research directions:

- Task grouping
- Transfer relationships
- Task embeddings

Task grouping:
The idea of the grouping-tasks approach is to avoid negative transfer between tasks by separating those tasks in the training process that will have a negative transfer between them. In this approach, the main body of research studies (e.g. [1], [5], [16]) use the approach of training all tasks individually and afterwards training them

jointly in pairs or groups of multiple tasks. Afterwards, the training process is compared between clusters that can be built out of tasks that improve their performance while being trained jointly and those that do not improve [5]. Using this approach, Alonso and Plank [1] found that auxiliary tasks with label distributions containing a high entropy but a low kurtosis are generally beneficial to train as auxiliary tasks to the main task. This result is strengthened by the findings of Bingel and Søgaard [5].

Doersch and Zisserman [16] did not use the trial-and-error approach used in the formerly discussed approaches. Instead, they used an algorithm called selective sharing (shown in Figure 15) for grouping the tasks into different clusters. This architecture uses the same input for all tasks, a shared feature extractor and a shared representation. Following the shared part of the architecture, task specific decoder heads predict a different output for each task. The gradients of the different tasks are used for building clusters of the tasks: The similarity between different gradient vectors is used as the predictor for task similarity. While training, the tasks are adaptively merged into clusters based on the gradients. When merging two tasks, the task-specific decoder heads are merged as well so that they share parameters. The algorithm stops at the point where the clusters no longer change. On the one hand, this approach is inexpensive and learns task features for understanding task relationships. On the other hand, it assumes that similarly behaving gradients go together with beneficial task combinations. This assumption breaks down more and more in the course of the training period [10].

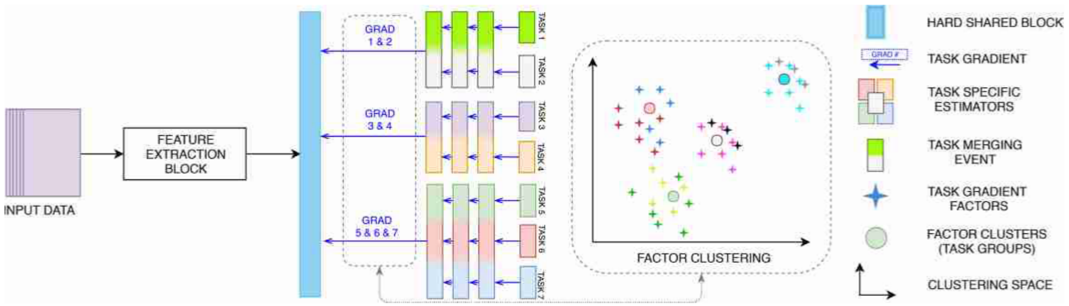


FIGURE 15: Selective sharing architecture [63].

Standley et al. [62] built a framework and reported an empirical study of task groups using the so-called Taskonomy dataset [83] (The Taskonomy dataset will be explained in detail later in this thesis). Furthermore, they divided a group of tasks into different clusters in which positive transfer existed between tasks in the same cluster. This was accomplished by using a branch-and-bound algorithm[2] to find the task clusters that benefit from joint training. We provide a more in-depth review of [83] in Chapter 3.1.

Transfer relationships:
The main idea of transfer relationship research is to learn how tasks should be grouped for joint learning. This approach differs from the task-grouping approach regarding the fact that it does not train the tasks simultaneously but in a two-step process, selecting an optimal source task (or multiple-source tasks) from which to

[2]A branch-and-bound-algorithm is a problem solving algorithm based on the assumption that the solution set can be divided into smaller subsets of solutions. [34]

transfer, then carrying out TL by moving the knowledge from the source to the target task for just those combinations that are worthwhile.
Zamir et al. [83] tried to learn task affinities using their Taskonomy dataset. For each pair of tasks, they asked the question "How well can we perform task i by training a decoder on top of a feature extractor which was trained on task j?" [10, p. 30]. This summary question is a simplification in the sense that j refers to not only a single task but also to multiple-task combinations. To find the ideal set of source tasks, the problem is modeled as a Boolean integer programming problem[3]. The result is visualized as a directed graph, while the tasks are represented as nodes. A directed edge between task i and task j means that task i is contained in the set of possible source tasks for task j (explained in more detail in Section 3.1; the graph is shown in Figure 19). The main drawback of the approach of [83] is that it is a brute-force approach and is therefore extremely computationally intensive.
Dwivedi and Roig [17] and Song et al. [59] introduced two much cheaper methods than that of [83].
Dwivedi and Roig [17] used RSA [4] [38] to compare two neural networks and build a measure of similarity. The underlying assumption is that tasks that have positive transfer will learn similar representations. By comparing the learned representations, it is possible to predict TLP without actually performing the transfer. [59] followed a similar approach to [17]; however, they used attribution maps. They compared the attribution maps of two networks on the same input. Their basic assumption was that tasks that complement each other in learning will pay attention to the same parts of an input image. The attention of the networks can be compared by using the attribution maps.
The work of Dwivedi and Roig [17] and Song et al. [59] is further analyzed in Sections 3.2 and 3.4.

Task embedding:
The task-embedding approach is a general form of learning task relationships. The main disadvantage of this method is that any new task to analyze must be directly related to the tasks already trained. It is mainly used in the area of meta learning. [10]

2.5 High-Resolution Network (Yannic)

A HRNet is a CV architecture specifically designed for CV tasks in semantic segmentation, pose estimation and object detection. It differs from other architectures in the way, that it processes input images throughout the whole network with the same resolution size. Commonly, architectures are designed to encode an input image to a smaller size by down-sampling the original size and only later decoding it back to its original form. Wang et al. [77] argue that re-establishment of the original size is characterized by information loss because information based on the original image size is not stored and lost during down-sampling and thus not completely recoverable during up-sampling at a later stage.
In addition to processing the original image resolution in the whole network, HRNet also contains parallel networks with smaller resolution sizes with a factor of 1/2 and 1/4 of the original image size. The whole process is visualized in Figure 16, and it

[3]A boolean Integer Programming problem is an optimization problem which can be solved by optimizing a linear function witch's variables are bound by a set of linear constraints. [10]
[4]Method drawn from Neuroscience and further explained in Section 3.2.

is characterized by the fact that all the networks communicate and exchange information with each other. Wang et al. [77] argue that each individual network scale recognizes different important features based on the resolution of the image and transmit this information to the other network scales and vice versa. Up-sampling and down-sampling are thus less prone to information loss because the information is stored at different scale instead of being partly deleted. At the end of the HRNet

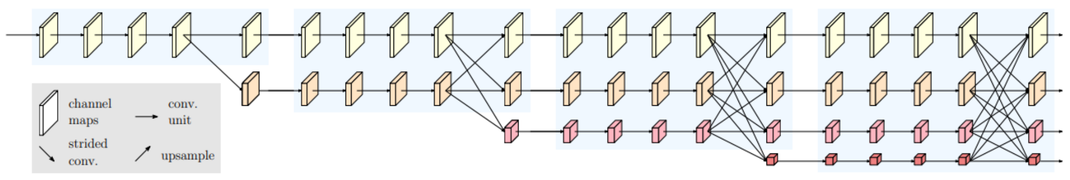


FIGURE 16: Abstract architecture of HRNet.

architecture, an output is produced at one specific network scale depending on the CV task. The end process is displayed in Figure 17 detailing the concatenation of all parallel networks to the specific network that outputs the result. Information from each network is passed in order to ensure scaling-based specific information in the output of the result. Wang et al. [77] argue that HRNet is suitable for use as a backbone network for semantic segmentation, object detection and human pose because it outperforms other state-of-the-art methods in each of these tasks.

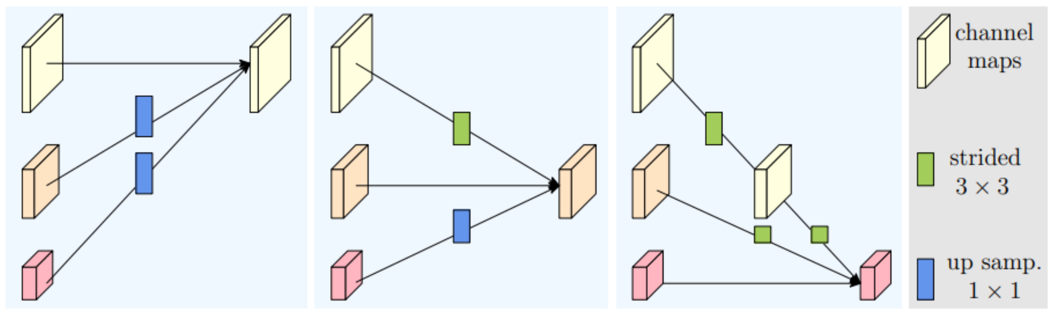


FIGURE 17: High-resolution heads of HRNet with one specific head, that outputs the result.

2.6 Mathematical Formulas (Yannic)

It must be said that our methodology and related literature can only be truly understood if certain mathematical principles are known. Therefore, important basics are presented and explained in this section.

2.6.1 Bayes Factor

The BF is used in each of our experiments as a comparison metric and one of the main drivers in deciding on one model. In research, the BF has been mentioned as a comparison technique between a null hypothesis and an alternative approach with the goal to defend one of the hypotheses against the other. As a model comparison metric, it is used for model selection by determining which model better fits the data.

The BF is used in order to calculate the posterior odds using the following formula [50]:

$$\frac{P(M0|data)}{P(M1|data)} = \frac{P(data|M0)}{P(data|M1)} * \frac{P(M0)}{P(M1)} \tag{2.3}$$

The mathematical part $\frac{P(data|M0)}{P(data|M1)}$ is highlighted as BF. Its outcome is described as a shift in belief that occurs when observing the model performance of Model 0 (M0) and Model 1 (M1) after new data is given to them. The ratio between the two models defines the preference of one over the other. The magnitude of preference K of one model over the other is categorized and counts as follows, when M0 is the numerator and Model 1 is the denominator [50, p. 158]:

- Grade 0. $K > 0$. M0 is supported
- Grade 1. $1 > K > 10^{-0.5}$. Small evidence against M0
- Grade 2. $10^{-0.5} > K > 10^{-1}$. Substantial evidence against M0
- Grade 3. $10^{-1} > K > 10^{-1.5}$. Strong evidence against M0
- Grade 4. $10^{-1.5} > K > 10^{-2}$. Very strong evidence against M0
- Grade 5. $10^{-2} > K >$. Decision based evidence against M0

2.6.2 R-Squared

R^2 is a method that follows a similar intention to the BF method in terms of model comparison and model selection. It is defined as the coefficient of determination and evaluates the quality of the fit of the line to the data in regression tasks. It uses the following equation [28]:

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{2.4}$$

where

1. RSS: Residual sum of squares, which calculates the difference between the ground truth value Y_i and the predicted value $\hat{Y}_i$ for each data point i
2. TSS: Total sum of squares, which calculates the difference between actual value Y_i and predicted mean value $\bar{Y}_i$ for each data point i

R^2 can take values between 0 and 1 in which a higher value means a better fit of the line and therefore a higher dependency between two variables, whereas a lower value means more independence between two variables [9].

2.6.3 Spearman Correlation

The Spearman correlation is defined as a rank correlation coefficient. We use the metric in the first two experiments in this work to evaluate our method against the benchmark. The correlation of the method is calculated by the ranks of the variable in each method. Spearman is used in bivariate statistics and measures the type and strength of the relationship between two variables through linearity. The mathematical formula is as follows [61]:

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n * (n^2 - n)} \tag{2.5}$$

where

1. r_s: Spearman rank correlation
2. d_i: difference between the ranks of corresponding variables
3. n: number of observations

The strength of relationship between two variables is represented by a value ranging from -1 to 1 with -1 meaning a negative correlation, 0 no correlation, and 1 a perfect correlation.[61]

2.6.4 Pearson Correlation

The Pearson correlation is a measure of linear correlation between two sets of data and outlines the strength of linear relationship between two variables. The following formula is used to calculate the Pearson correlation coefficient (PCC) given two random variables X and Y [76]:

$$PCC(X,Y) = \frac{cov(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X-\bar{X})(Y-\bar{Y})]}{\sigma_x \sigma_y} \tag{2.6}$$

where:

1. $\bar{X}$ is the average value of X
2. $\bar{Y}$ is the average value of Y
3. σ_x is the standard deviation of X
4. σ_y is the standard deviation of Y

The Pearson correlation can take values ranging from -1 to 1, resulting in the strength of relationship between two variables. A value of -1 means a negative correlation between two variables, while 0 means no correlation and 1 a perfect correlation.[36]

2.6.5 Cosine Distance Function

The cosine distance function is used to get the dissimilarity between two representations by calculating the angular distance between two vectors, which are projected in a multi-dimensional space. The mathematical formula is hereby as follows [21, p. 55]:

$$cosine\_distance = 1 - cosine\_similarity = 1 - \frac{A * B}{||A|| * ||B||} = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.7}$$

where:

1. A: Vector A
2. B: Vector B
3. $||A||$: Euclidean norm of vector A
4. $||B||$: Euclidean norm of vector B

Chapter 3

Related Literature

In this chapter, we introduce the current state of literature in the field of TL model selection. In Section 3.1, we discuss the work of Zamir et al. [83] in which the authors create a task taxonomy that serves as a ground truth of TLP in current state-of-the-art methods for predicting TLP. These state-of-the-art methods for TLP prediction are explained in Sections: 3.2, 3.3 and 3.4.

3.1 Taskonomy - Disentangling Task Transfer Learning (Daniel)

With this work, Zamir et al. [83] propose a fully computational approach for modeling the structure of the space of visual tasks. Fully computational refers to the fact that the authors perform the full computation of transfers using a brute-force approach, meaning that they do not use a prediction or heuristic but create the ground truth. With this ground, truth they create a benchmark against which to evaluate TL prediction methods.
The two main contributions of Zamir et al. [83] are as follows:

1. They create a dataset of task similarities called Taskonomy.
2. They propose a framework for creating a computational taxonomic map for task TL.

Taskonomy is an image dataset containing four million images. The images are of indoor scenes from more than 600 buildings and cover 26 different CV tasks, which are all completely labeled. The 26 CV tasks are in the areas of 2D, 2.5D, 3D and semantic tasks. The complete pixel-level geometric information is provided using meshes, and the semantic annotations are generated as pseudo-semantic annotations by applying knowledge distillation [84].
The goal of the framework is to make visible the underlying and hidden task space structure of visual tasks. Furthermore, it aims to provide a framework that can be used to map the space of visual tasks [84]. This task space structure [84] refers to the computationally found relations between tasks; it can be rephrased as which tasks benefit from joint training (positive transfer in TL) and which do not (negative transfer in TL). The authors formulate the problem as follows: a set of target tasks τ with $\{t_1, ..., t_n\}$, a set of source tasks S with $\{s_1, ..., s_n\}$ and the supervision budget γ (the supervision budget is equal to the maximum number of trainable source tasks, for example due to computation restrictions).
The task dictionary ν is defined as the joint set of source and target tasks: $\nu = \tau \cup S$. In the analysis, the four tasks colorization, jigsaw puzzle, in-painting and random projection are source-only, which means that they are only used as a source for TL and never as the target of the transfer. The source-only tasks are given as $S - S \cap \tau$.

The results are achieved in a four step process shown in Figure 18:

1. Train a specific network for each task.
2. Perform transfers from different orders (single source, double source, etc.).
3. Normalize the results.
4. Solve subgraph selection and build a hypergraph.

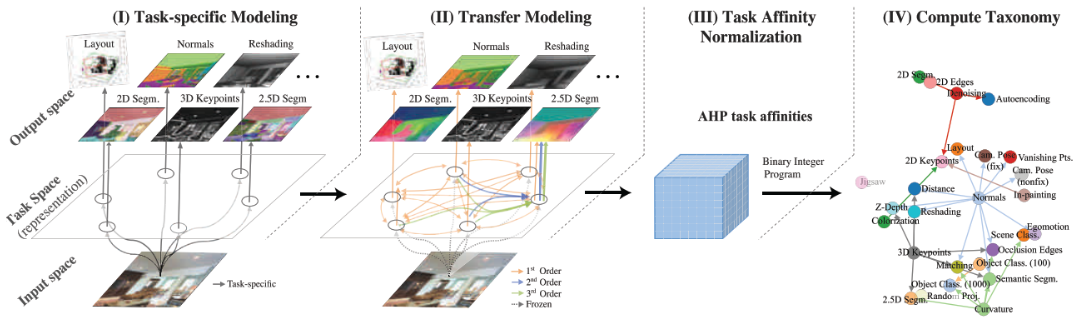


FIGURE 18: Taskonomy four step process framework [84].

In the first step, a fully supervised model is trained for each task. For this, every task uses the same encoder (ResNet-50 [29] without pooling). The decoders need to be different since the tasks have different output requirements.
In the second step, multiple transfers are done from all $s_i \in S$ to all $t_j \in \tau$. Furthermore, higher order transfers are done using multiple tasks as source tasks. Since the number of possible transfers explodes combinatorially (powerset $\mathcal{P}(\mathbb{S})$), the beam-search algorithm[1] is used to prune down the number of transfers with multiple sources and select only the most promising combinations.
In the third step, the results of the transfers are normalized using an analytical hierarchy process (AHP)[2].

[1]Beam search is an hearistic graph based search algorithm. It is a variation of breadth-first-search and uses a beam width which specifies how many of the top branches should be saved to memory for further exploration. [4]

[2]AHP is a "methodology for structuring, measurement, and synthesis" [23, p. 469]. It compares objectives in a pairwise manner. The method contains three primary functions: structuring complexity, measurement and synthesis. [23]

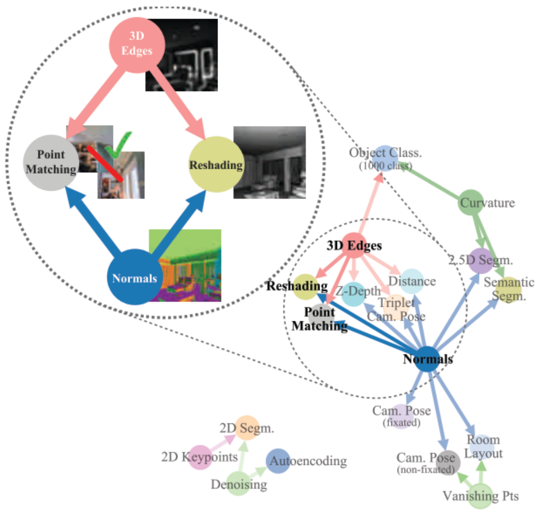


FIGURE 19: Taskonomy example of a hypergraph [84].

In the fourth step, the subgraph selection problem is solved using binary integer programming (BIP)[3], and a hypergraph is constructed. Figure 19 shows a created hypergraph as an example. In the graph, each node is a task. An edge from node i to node j implies that there exists a feasible transfer from task i to task j. The thickness of the edge demonstrates the performance of the transfer (the thicker the edge, the better the performance of the transfer).
In [84], the authors perform 3,000 knowledge transfers from source to target tasks with over 47,886 graphics processing unit (GPU) hours of training time. To analyze the generalization of the found task structure (called Taskonomy) the authors analyzed an all-for-one scenario, which means that just one target-only task is used. Target-only means that the task is never used as a source task ($|\tau| = 1$ and $|S| = n - |\tau| = n - 1$), which is a typical usecase. Since the task is target-only, no task-specific network is trained for it. Only 16,000 images were used for training a transfer net (TN) based on the source tasks. With this procedure, the authors aimed to localize the target task inside the Taskonomy graph.
The main disadvantage of the Taskonomy brute-force approach is its high computational cost. Performing the transfers is very resource intensive. Furthermore, for every task, many human notations are necessary as the ground truth for training, which is costly as well. Another problem is that when a researcher wants to analyze a new task that is not already contained in the graph, new transfers have to be trained between the new task and all the tasks already included in the graph. Therefore, the runtime grows polynomially with $\mathcal{O}(n^2)$ with the number of tasks involved (only with respect to the first order transfers) [59].

3.2 Representation Similarity Analysis for Efficient Task Taxonomy and Transfer Learning (Yannic)

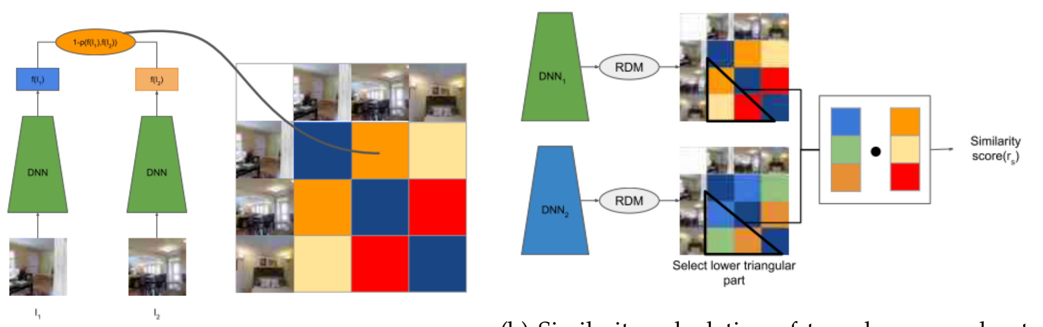
The RSA approach was originally used in the neuroscience to calculate the similarity between three different areas (brain-activity measurement, behavioral measurement and quantitative computational modeling) by measuring the correlation between brain data responses and computation or behavioral models [38][17].

[3]BIP is a mathematical optimization program in which all variables are boolean integers (0 or 1). It is NP-complete.

This RSA concept was adopted in CV [17] for a model-to-model similarity-comparison approach in two applications, namely task taxonomy and TL. The following points were investigated regarding the RSA approach:

- Task relationships between different tasks according to their similarity
- Size of model and size of image-set for adequate similarity measure
- Model-selection method in CV for TL

Dwivedi and Roig [17] adopted the methodology of RSA using it in combination with RDMs to measure the similarity between different CV tasks. A RDM stores for each respective visual task the dissimilarity between different representations, that a task-based model outputs. These task-based RDMs are then compared with one another by calculating the Pearson correlation between them, thus displaying the similarity between the tasks. A higher similarity between the tasks is interpreted as more suitable for TL. Dwivedi and Roig [17] describes the Taskonomy approach in [84] as unsustainable due to the high degree of computational resources and training time needed. The authors mention, that each task from a set of pretrained tasks needs to be trained from scratch every time, if they want to calculate the TLP of a new task. The current state-of-the-art approach in the real world is also described as incomplete, because it ignores and therefore probably overlooks pretrained models that are more suitable for the target task. Dwivedi and Roig [17] claim that their method overcomes both issues as it is more efficient and faster. The authors validate their approach by comparing it to the TLP achieved and published in the Taskonomy experiment [84].

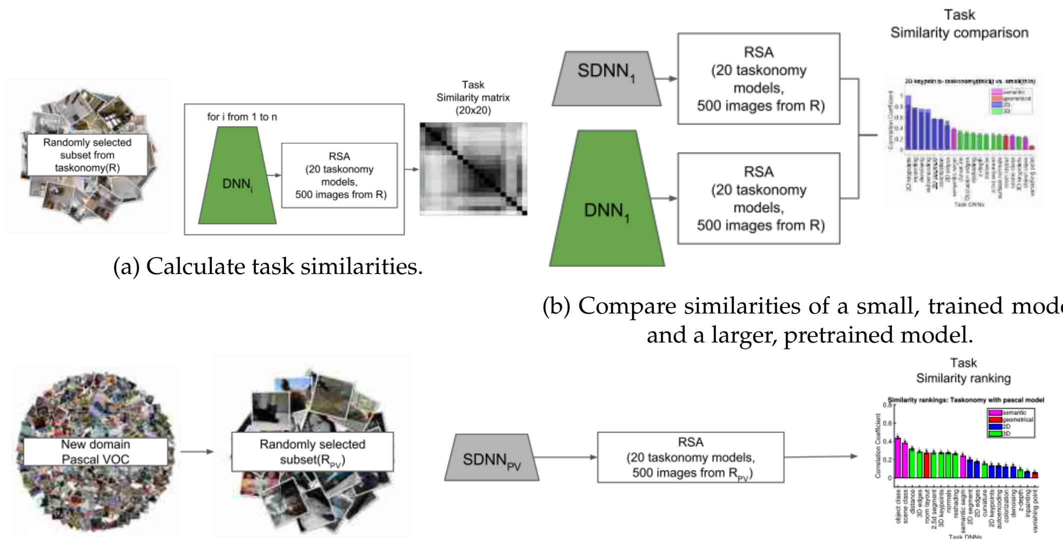


(a) RDM´s of two deep neural networks.

(b) Similarity calculation of two deep neural networks.

FIGURE 20: RSA-method [17].

Dwivedi and Roig [17] argue, that their approach bypasses the previously mentioned issues in an efficient manner, because their method does not need any model training but only one forward pass and a small amount of data for the visualization of task relationships. The RSA method is partitioned into two steps. A RDM for each model is first obtained by using the representations of input images at a certain layer through a forward pass. The representations are then used to calculate a dissimilarity score between each pair of representations with the formula $1 - Pearson\_correlation$. Both steps are illustrated in Figures 20a, and Figure 20b. After having obtained task-based RDMs, the similarity between the RDMs is obtained through the lower or upper triangular part of the RDMs and the Spearman correlation [17]:


Randomly selected subset from taskonomy(R)
for i from 1 to n
DNN$_i$
RSA (20 taskonomy models, 500 images from R)
Task Similarity matrix (20x20)
SDNN$_1$
DNN$_1$
RSA (20 taskonomy models, 500 images from R)
Task Similarity comparison
New domain Pascal VOC
Randomly selected subset(R_{PV})
SDNN$_{PV}$
RSA (20 taskonomy models, 500 images from R_{PV})
Task Similarity ranking
Similarity rankings: Taskonomy with pascal model
Correlation Coefficient
semantic
geometrical
2D
3D
Task DNNs

(a) Calculate task similarities.

(b) Compare similarities of a small, trained model and a larger, pretrained model.

(c) Small pretrained model for new computer vision task in transfer learning.

FIGURE 21: RSA-approaches in [17].

Dwivedi and Roig [17] analyzed the usage of RSA in three areas, which are displayed in Figure 21. They concluded, that their RSA approach clusters similar tasks into three groups as it is the reported in the Taskonomy paper explained in Section 3.1, namely in 2D, 3D and semantic tasks. It is also described that the similarities between different tasks started to decrease when measured deeper in the network. As a second result, they found that their method shows a high correlation with a smaller model trained with the same number of images as with a larger network in Taskonomy, which means that a smaller dataset promises the same transfer learning results as is the case with a large dataset. Furthermore, the authors discovered, that the RSA similarity score had a high correlation with the TLP of the Taskonomy approach. RSA seems to be an attractive alternative to the Taskonomy approach, due to its good experimental results and its computational efficiency. Dwivedi and Roig [17] also highlight the flexibility of the method and its potential usage in other areas like MTL:

> RSA can be used for deciding different branching out locations for different tasks, depending on their similarity with the representations at different depth of the shared root. [17, p. 3]

3.3 Duality Diagram Similarity: A Generic Framework For Initialization Selection In Task Transfer Learning (Daniel)

In Kshitij Dwivedi et al. [40], the authors propose a general framework called duality diagram similarity (DDS) that helps selecting the optimal model initialization for TL from a set of different source models. DDS is highly correlated with TL performance. Furthermore, the DDS framework is useful for selecting the location in a model: the specific layer from which to transfer. The general motivation in Kshitij Dwivedi et al. [40] is to discover how one should select a model to achieve the highest possible TLP on a new task when different pretrained models are available.

The DDS method is based on duality diagram (DD)s[4] introduced in Escoufier [18]. With DDS, it is possible to compare the representations of two deep neural network (DNN)s. After comparing the DNNs, DDS predicts the TLP when a transfer is done from one DNN to the other. DDS is formulized as follows:
A set of encoders of already trained tasks is given. The goal is to compare the similarity of each of the pretrained tasks with the target task to create a ranking of predicted TL-performance.
In the following, X refers to one of the pretrained tasks, while Y refers to the target task: X and Y are matrices containing the features of the pretrained task and the target task. These features are created by passing n images (feed forward) into the DNNs and extracting the features for every image. The matrix D contains weights for the images, while the matrices Q_X and Q_Y store relations between the feature dimensions of the pretrained task and target task, respectively.
Figure 22 visualizes the method of [40]. The process works in three steps:

1. Transform the data.
2. Generate pairwise distance maps.
3. Compare distance maps.

In the first step, the features X (or Y) in combination with D and Q_X (or D and Q_Y) are used to create the DD $\hat{X}$ (or $\hat{Y}$) from D, X and Q_X: $\hat{X} = DXQ_X$ (the same for $\hat{Y} = DYQ_Y$).
In the second step, a similarity function f is used to create the pairwise similarity matrices M_X and M_Y by giving $\hat{X}$ and $\hat{Y}$ as input to f.
Third, the similarity function g returns the similarity S of M_X and M_Y (Equation 3.1) (the index i in Equation 3.1 refers to the source task from the set of pretrained tasks):

$$S_i = g(M_X, M_Y) \tag{3.1}$$

Kshitij Dwivedi et al. [40] experimented with different version for the similarity function f. They used the functions linear, Laplacian, RBF, Pearson, Euclidean and cosine. They report that the DDS combination containing Z-score[5] for Q and D and cosine function for f is the best option. In this work, Z-scoring is applied to D, X and Q: $D = I_{nxn} - 1_{nxn}/n$, $X = X_{nxchw}$ and $Q = S_{chwxchw}$, where I is the identity matrix; the matrix is filled with 1s, and X is the output feature map extracted from a feed forward pass (with c being the number of channels, h the height, w the width and n the number of images).
Using Z-score Spearman's correlation outperforms Pearson's correlation as the function g (see on Figure 22) for comparing the similarity matrices (g).

[4]The DD is a method based on the Principal Component Analysis, which can additionally deal with changes of scale, variables, weighting of statistial units (for this work: feature representations), and decentering of representations.[18]

[5]Z-scoring is a standardization method, using the mean μ and the standard deviation σ. The formula for the z-score is $z = \frac{(x-\mu)}{\sigma}$

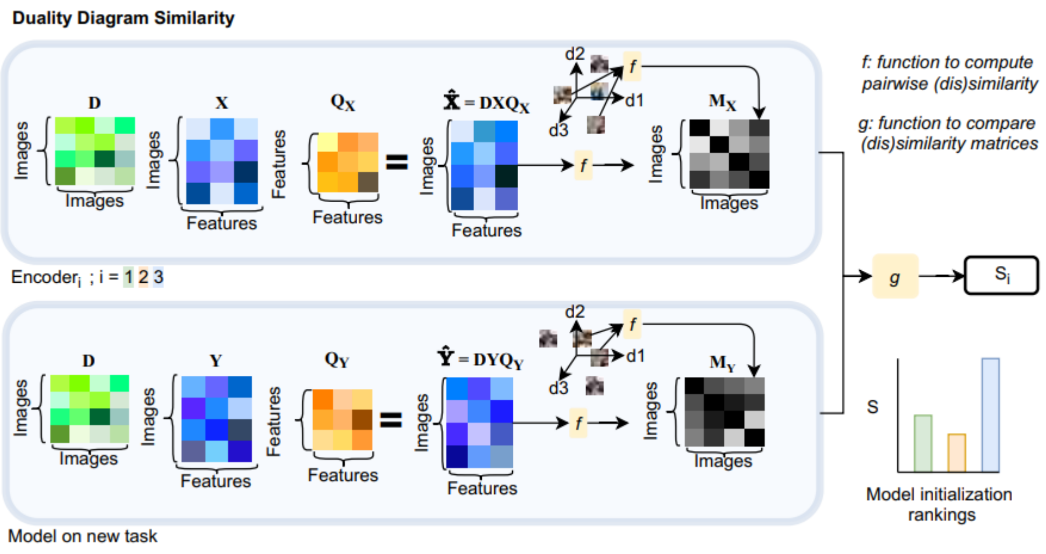

Duality Diagram Similarity
D
X
Qx
X̂ = DQX
Mx
Images
Features
d1
d2
d3
f
f: function to compute pairwise (dis)similarity
g: function to compare (dis)similarity matrices
Encoderi ; i = 1 2 3
g
Si
Y
QY
Ŷ = DYQY
MY
S
Model initialization rankings
Model on new task

FIGURE 22: duality diagram similarity framework [40].

Kshitij Dwivedi et al. [40] compared DDS using cosine-similarity with DDS using Laplacian as the kernel function f with the results of state-of-the-art methods: Taskonomy (as ground truth), attribution maps [59] (explained in detail in Section 3.4), DeepLIFT [59], ϵ-LRP [59] and RSA [17] (explained in detail in Section 3.2).
The authors [40] compared the DDS method with the results of the Taskonomy approach [84] on a basis of 17x17 task transfers. They achieved a correlation of 0.86 with the Taskonomy ground truth in less than two minutes. With these results, the authors report a 10% increase in performance compared with state-of-the-art methods, and the method is several magnitudes faster than the brute-force approach of [84].

3.4 Deep Model Transferability From Attribution Maps (Daniel)

The approach of Song et al. [59] uses the attribution maps of DNNs to estimate transfer ability. To accomplish this, it projects the DNNs into a model space in which the points in the space refer to networks; the distance between points gives the measure of relatedness between two DNNs and is calculated by the deviations of their produced attribution maps.
The model space is defined as a space on top of the attribution maps where distance between points represents the distance between produced attribution maps [59].
The general assumption of this approach is that models that focus on similar regions of input images are expected to produce correlated representations, and correlated representations give rise to favorable transfer learning results.
The authors specify their problem setup similarly to the problem setup of Taskonomy in Section 3.1:
$M = \{m_1, ..., m_N\}$ is a set of pretrained models, in which N is the number of models involved. t_i is the task treated by model m_i and the task dictionary τ is given by $\tau = \{t_1, ..., t_N\}$. It is assumed that no labeled annotations are available, and the goal is to

> efficiently quantify the transfer ability between different tasks in τ, so that given a target task, we can read out from the learned transfer ability

matrix the source task that potentially yields the highest transfer performance. [59]

The approach involves a three step process that is visualized in Figure 23:

1. Create probe-dataset.
2. Compute attribution maps.
3. Estimate model transfer ability.

In the first step, a probe-dataset is created. This is done with unlabeled data, so no human supervision is necessary. For this, a random selection of 1,000 images is taken. The probe-dataset is the same for all tasks. The idea of the probe-dataset is that, later on, the responses of the different DNNs are compared to the same stimuli (the stimuli that is the same for all DNNs is the probe-dataset).
In the second step, the attribution maps are created. Three different attribution methods are used: saliency map[6], DeepLIFT[7] and $\in$-LRP[8].
In the third step, model transfer ability is estimated. This is accomplished by calculating the cosine-similarity between the averaged attribution maps.
In comparison to Taskonomy the attribution map approach was shown to be faster and cheaper in terms of computational requirements. Song et al. [59] report their approach to be in the factor of $\frac{EN(T-1)}{M}$ more efficient. E refers to the epochs trained on the training data which has the size N, while T is the size of the task dictionary and M is the size of the Probe-dataset. Simultaneously, the attribution map approach produces a highly similar tansferability topology to Taskonomy. Therefore, the authors propose the attribution map approach as a competent transferability estimator that is especially useful when the task dictionary is large or frequently updated so that recalculation is necessary.

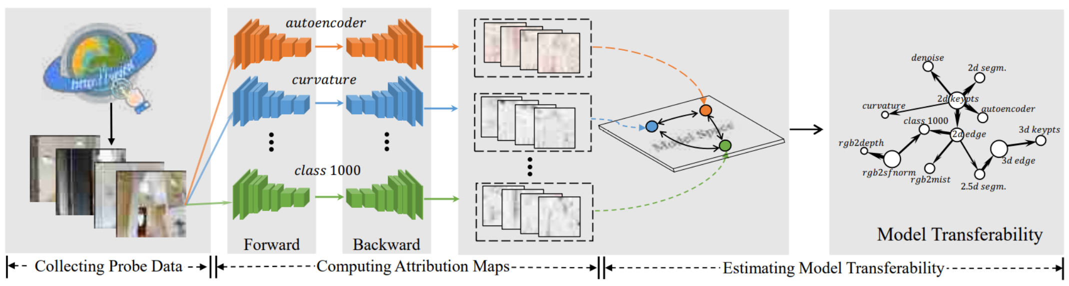


FIGURE 23: Three-step process of model transferability graph creation using attribution maps [59].

The advantages of these methods are that there are no constraints on the architectures of the DNNs and no human annotations are required. Furthermore, updating the model space is inexpensive since it only requires calculating the nearest neighbors in the model space, which is much cheaper than updating in Taskonomy.

[6]The attribution map saliency map calculates the absolute value of the partial derivative of the output for a given input. [59]
[7]DeepLIFT is an improvement of the saliency maps method. It multiplies the gradient with the input signal: This is equal to a first-order Taylor approximation of how the output would change when the input is fixed to zero [55]
[8]$\in$-LRP"computes the attributions by redistributing the prediction score (output) layer by layer until the input layer is reached." [59, p. 4]

Chapter 4

Many-To-One Task Similarity (Daniel)

In Chapter 3, we discussed current methods targeting the question how to find the best pretrained model for doing transfer learning for a selected target task?
Zamir et al. [83] present a fully computational approach that gives the most precise results. However, it is highly cost intensive and therefore not feasible for many applications. Song et al. [59] use attribution maps for predicting the TLP, while Dwivedi and Roig [17] and Kshitij Dwivedi et al. [40] use feature map correlations of source and target task as their basis for predicting TLP. All the mentioned methods based on attribution maps and feature representations are several magnitudes cheaper than the fully computational Taskonomy approach. However, they have one major drawback: None of them can use the combined knowledge of multiple sources for the TLP prediction.

With this thesis, we propose a new method we call Many-To-One Task Similarity (MOTS). With MOTS, we tackle the possibility of using multiple source tasks for the TLP prediction. MOTS is a method for predicting TLP and the similarity between source and target tasks based on a couple of given source models. MOTS takes one or multiple source-task models and a target task as input and ranks the source models in terms of TLP when using them as the source. MOTS is mainly built on the work of Kshitij Dwivedi et al. [40] and Dwivedi and Roig [17]. However, the main difference is that it can use multiple source tasks as sources for the transfer. Figure 24 presents a general overview of the method.

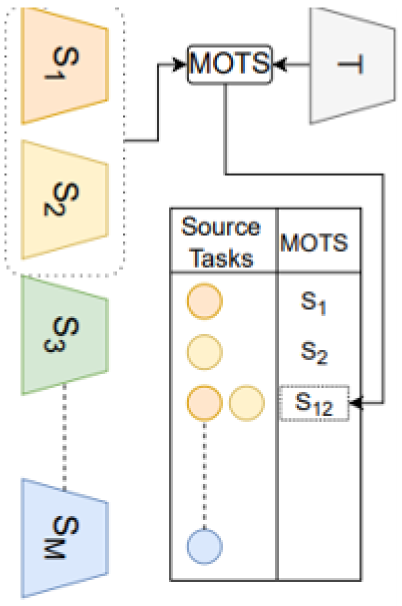


FIGURE 24: **MOTS ranking generation.**The representations of different source-task combinations are compared by MOTS to representations of the target task. MOTS returns a ranking of all source combinations for the provided target task. The source combinations are ordered by their predicted transfer learning performance.

MOTS uses the extracted feature maps of all source tasks and the feature map of the target task. It compares the similarity of source and target task by creating RDMes out of the features and compares these RDMs to find similarities. The prediction of TLP comes from the R^2-value of a linear regression from source RDMs to the target RDM.
The MOTS process in detail is the following:

1. In the first step, the activations/feature maps of all source models need to be extracted. This is done by forwarding N images through the network and extracting the feature maps of the layer from which to transfer from. Having this, an RDM of dimension N is calculated from the feature maps individually from each model. For the distance function k, we use the cosine function because Kshitij Dwivedi et al. [40] identify it as the most robust. These steps need to be performed for all source tasks and the target task to create an RDM independently for each model (see Figure 25).

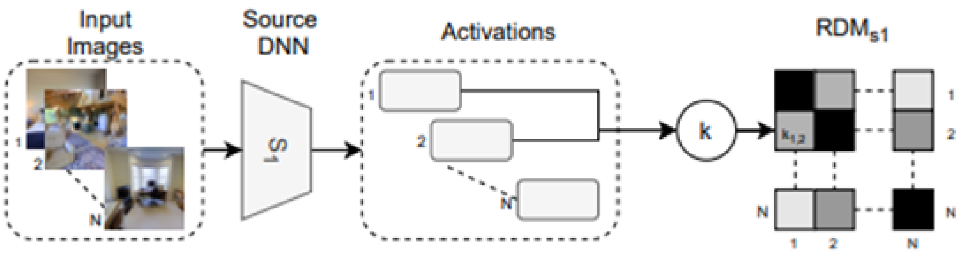


FIGURE 25: Step 1: RDM creation from feature maps.

2. In the second step, a set of training images is used to estimate the weights for a linear regression using the RDMs of the source models as independent

variables and the RDM of the target task as the dependent variable (See Figure 26).

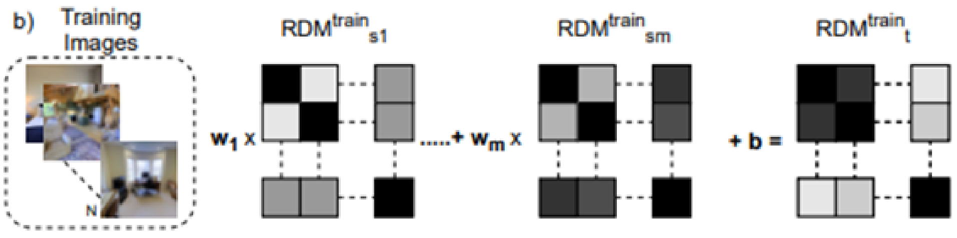


FIGURE 26: Step 2: Fit linear regression.

3. In the third step a set of test images is used to evaluate the quality of the fit by calculating the R^2 value. (See Figure 27)

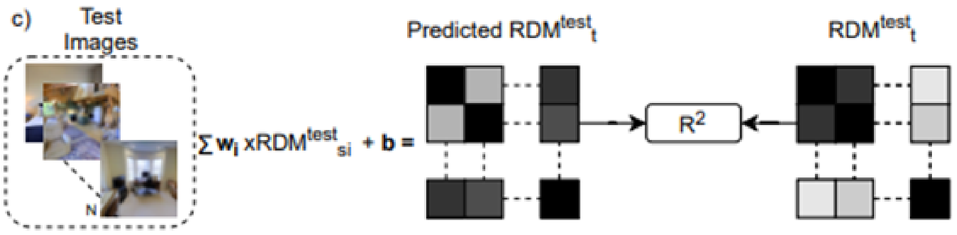

c) Test Images
N
Predicted RDM^{test}_{t}
RDM^{test}_{t}
$\sum w_i$ xRDM^{test}_{si} + b =
R^2

FIGURE 27: Step 3: Extract measure of quality of fit (R^2).

For evaluating MOTS, we use a ground truth that gives the actual TLP of the transfers from each source task. Figure 28 displays the evaluation of MOTS against a provided ground truth. After creating the ranking of all source combinations, the MOTS rankings are compared to the actual TLP as the ground truth. The comparison between both rankings is done using Spearman's correlation (see Figure 28.

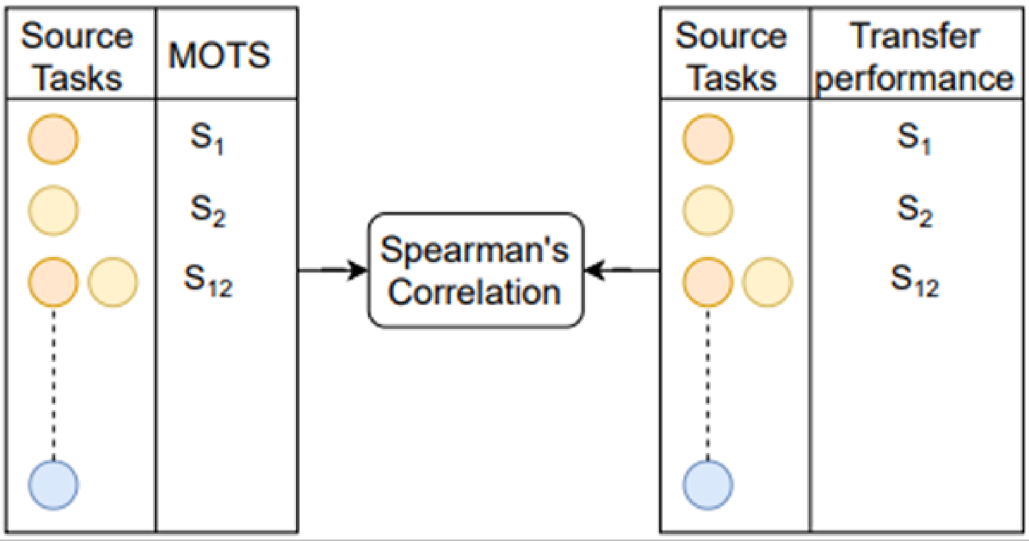


FIGURE 28: MOTS evaluation by comparing MOTS ranking with ground truth ranking of real transfer learning performances.

Chapter 5

Experiments and Results

This chapter presents the implementation of MOTS, the experiments we performed and the results of each experiment. Our experiments using MOTS for single-source transfers and multi-source transfers as well as a combination of both and the comparison with the ground truth and other state-of-the-art methods are described in Section 5.3. We benchmark the results of MOTS against other state-of-the-art results, that use the same ground truth for comparison.

5.1 Datasets (Yannic)

Taskonomy:

This dataset was created for the deployment of the Taskonomy experiments in [84]. It consists of a variety of indoor scenes with the following statistics of the dataset:

- 4.5 million scenes
- 600 buildings
- 25 tags per image
- 1,024 resolution for taxonomy and transfer learning tasks.

The Taskonomy dataset includes annotations for every tasks on every image and makes it therefore suitable in a TL context as ground truth data according to Zamir et al. [84]. They argue that through the annotations for each image on each task, task intrinsics play a major role in TLP, less so the characteristics of each pixel of an image, when a task is being trained on them and then used for TL. The TLP is thus not distorted as it could be, when using the second approach. The TL results show therefore a more significant measure in the way this dataset was built.
We use the Taskonomy dataset for our experiments in chapter 5.3 to compare the performance of our MOTS method with other state-of-the-art similarity methods.

Pascal VOC Semantic Segmentation:

The Pascal VOC dataset consists of outdoor scenes with the aim to offer different images in terms of variability to avoid a bias towards something specific inside of an image when the pattern occurs in multiple images. In terms of variability, this dataset provides the following characteristics according to [20][19]:

- object size
- orientation

- pose
- illumination
- position
- occlusion

The Pascal VOC 2012 dataset includes 20 object categories of pixel-accurate segmentation annotations, bounding box annotations and object class annotations. Pascal VOC was created with two intentions according to [20][19]:

- to offer a publicly available dataset with ground-truth annotations to measure the performance of self-created algorithms
- to hold an annual competition with five challenges ranging from classification to detection, segmentation, action classification and person layout and dataset annotations specific to these tasks with the goal of new innovative implementations and creations of state-of-the art methods in the respective categories.

We used the annotated segmentation dataset for our experiment. The Pascal VOC 2012 segmentation dataset consists of 1,464 training images and 1,449 validation images [49].
We used the dataset on our first two experiments to compare the performance of our method MOTS with other state-of-the-art similarity methods and thereafter for our last experiment in Section 5.4 to validate our approach with a MTL architecture.

NYUD:

The so-called NYU-depth (NYUD) dataset consists of indoor scenes, that are extracted from video sequences and recorded by the RGB and depth cameras from Microsoft Kinect [56]. There are currently two versions of the dataset available: NYUD version 1 (v1) [56] and NYUD version 2 (v2) [57]. NYUD v1 covers 12 object categories in 2,347 frames extracted from different environments and manually labeled with the goal to combine depth maps intensities and dense labels of the images [56]. The NYUD v2 dataset is a new version of NYUD v1 that contains the following:

- 1,449 RGB and depth images
- 464 diverse annotated indoor scenes across 26 scene classes

Each image consists of a labeled class and an instance number for objects being in the same class. Furthermore, support annotations are included, which are represented in a triplet form shown by Silberman et al. [57]:

$$[Ri, Rj, type] \tag{5.1}$$

where:

1. R_i is region ID of the supporting object
2. R_j is the region ID of the supported object
3. $type$ indicates whether the support is from below (e.g., cup on a table) or from behind (e.g.,picture on a wall).

The v2 dataset was created to handle messy and untidy scenes with two goals:

- to disassemble each characteristic inside of a room, which are typically floor, walls, supporting surfaces and object regions, and
- to retrieve support relationships between the characteristics of a room.

We used NYUD v2 in Section 5.4 to validate our MOTS approach and the transferability in a new scenario with a modern MTL-architecture.

5.2 Hardware Used for Experiments (Daniel)

The training of neural networks is highly resource intense in the area of DL and even more so with the usage of images or video data. Therefore, it is necessary to have sufficient hardware availability. Especially the GPU is important since GPUs are made for processing many small computations in a parallel manner [53]. In neural networks in each forward pass, all calculations from each neuron of one layer can be done in parallel. Therefore, GPUs can greatly increase the training speed in comparison to central processing unit (CPU) training. Since most of the experiments in this thesis could not be run locally on our machines (or simply would take too long), we used two Ubuntu servers. The network hostnames of the two servers are g5 and g4.

g5-server:
The g5 is a Linux Ubuntu server using Linux kernel version 4.15.0-74-generic. It is a 64 bit version with 754 GiB of memory. The CPU is an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz with 16 physical CPU cores. The g5 has four NVIDIA Tesla V100 SXM2 32GB as GPUs that contain about 32,510 MiB GPU memory and a maximum power capacity of 300 watts each. CUDA version 10.2 is installed.

g4-server:
The g4 is a Linux Ubuntu server using Linux kernel version 5.4.0-72-generic. It is a 64 bit version with 1 TiB of system memory. The CPU is an AMD EPYC 7452 32-core processor with 32 physical cores; g4 has four A100-SXM4-40GB as GPUs that contain about 40,536 MiB GPU memory and a maximum power capacity of 400 watts each. CUDA version 11.2 is installed.

All our code was run in a single GPU manner. However, we used the different GPUs for different calculations in parallel.

5.3 MOTS in Comparison with Other State-Of-the-Art Methods

5.3.1 Experimental Setup (Yannic)

Ground Truth:
We used the TL results from the Taskonomy experiment [84] as the ground-truth data for comparison using our similarity method. We obtained the ground-truth file [1] from the GitHub page of Taskonomy in [66].
For the execution of our experiment on the Pascal VOC dataset, we used the ground truth from the DDS experiment [40], which is provided in the respective GitHub

[1]Ground truth files are downloadable at https://github.com/StanfordVL/taskonomy/tree/master/results/affinities . We use the ground truth file affinities.pkl

page of DDS [2] and consists of information about the TLP regarding the target task Pascal VOC semantic segmentation.

Tasks:
We used Taskonomy tasks from the Taskonomy experiment conducted by Zamir et al. [84] and described in 3 with the goal to research task-relationships between 26 tasks, ranging from 2D, 2.5D to 3D as well as semantic tasks. In the Taskonomy experiment, the TLP was calculated from a variety of different source tasks to one target task based on the 26 Taskonomy tasks. With respect to these experiments, we used the TL results of all source-task combinations to one target task as the ground truth, which included 471 transfers in different task combinations. The tasks we used are described in Table 3. We choose these tasks to make MOTS comparable to other state-of-the-art methods, especially to the current best performing similarity method DDS [40]. We obtained the ground-truth transfer results from a variety of different task-combinations, which were either transfers from one source task to one target task or two source tasks to one target task.
For the execution of our experiment on the Pascal VOC dataset, the same 17 Taskonomy tasks from Table 3 were used as source tasks to do single-source transfers to the Pascal VOC semantic segmentation task, which gave us in total 17 TL results. The ground-truth TL results included single-source transfers from each of the 17 Taskonomy tasks to the target task Pascal VOC semantic segmentation.

Feature Maps:
We used feature representations to build our RDMs by calculating the pairwise dissimilarity between each feature map. Resulting from this, we obtained task-based RDMs for each task described in Table 3. We used in total 5,000 feature maps that we found in the DDS experiments conducted by Dwivedi and Roig [17]. [3]

[2]Ground truth informations are stored inside a list under the following link: https://github.com/cvai-repo/duality-diagram-similarity/blob/master/jupyter_notebooks/DDS_vs_transferlearning(Pascal).ipynb

[3]Feature maps are downloadable at the GitHub repository of DDS in https://github.com/cvai-repo/duality-diagram-similarity/tree/master/features
We use the following files:

- taskonomy_pascal_feats_taskonomy_5000.pkl for 5,000 feature maps from the Taskonomy dataset
- taskonomy_pascal_feats_pascal_5000.pkl for 5,000 feature maps from the Pascal VOC dataset

Task
Autoencoding
Curvature Estimation
Denoising
Edge Detection (2D)
Edge Detection (3D)
Keypoint Detection (2D)
Keypoint Detection (3D)
Reshading
Depth Estimation
RGB2Mist
Surface Normal Estimation
Room Layout Estimation
Segmentation, Unsupervised(2D)
Segmentation, Unsupervised(2.5D)
Vanishing Point Estimation
Segmentation, Semantic
Classification, Semantic (1000-classes)

TABLE 3: Taskonomy tasks used in our experiment.

5.3.2 RDM Configuration (Yannic)

We examined different RDM sizes and different train-test split procedures in order to find the optimal size and the optimal RDM train-test split, that gives MOTS the best overall result. For each task-based RDM, we used z-standardization as our normalization function and cosine distance metric. There are four different RDM-train-test split methods for the linear regression of MOTS in which the size of the RDM test set is chosen to be equal to the size of the RDM training set or fixed at a size of 500 images. Training and test sets are either completely identical and called “train-train", meaning they use identical feature maps, or completely unequal and called “train-test", meaning the training set and test set are based on different feature maps. We use the term growing, which means, that both train and test-set increase in size. We as well use the term fixed, which means, that the RDM size always remains the same. We selected RDM sizes and train-test split procedures using the following logic:

1. RDM train-test split, where train set $=$ test set and train and test-sets are growing.
2. RDM train-test split, where train set $\neq$ test set and train and test-set are growing.
3. RDM train-test split, where train set $\neq$ test set and train-set is growing and test-set is always fixed at a size of 500 same images.

Table 4 shows our results for Cases 1 and 2, and Table 5 shows our resulting RDMs for Case 3. We executed MOTS on each of these predefined sizes and selected the one that performs the best in comparison with state-of-the art methods.

RDM train size	50	200	400	600	800	1,000	1,200	1,400	...	2,400
RDM test size	50	200	400	600	800	1,000	1,200	1,400	...	2,400

TABLE 4: RDMs, that we use for the prediction of our target RDM and are split into a train and test RDMs. So, when we use a train RDM with 50 images, we also use a test-RDM with 50 images. When we use a train RDM with 200 images, then we use a test RDM with 200 images, and so on. We use this so-called growing approach for the RDM train-test split procedure in which the train and test RDMs consists of different images; for the case of train and test being equal, the train RDM and test RDM use the same images.

RDM train size	500	1,000	1,500	2,000	2,500	3,000	3,500	4,000	4,500
RDM test size	500	500	500	500	500	500	500	500	500

TABLE 5: RDMs, that we use for the prediction of our target RDM and are split into a train and test RDMs. We use a fixed test-set size of 500, whereas the train RDM increases in terms of the number of images included. So, when we use a train RDM with 500 images, then we also use a test RDM with 500 images. When we use a train RDM with 1,000 images, we still use a test RDM with 500 images, and so on. We use this so-called fixed approach for the RDM train-test split procedure, in which train and test-RDMs consists of different images, and the test RDM always uses the same 500 images.

5.3.3 Data Preparation and Pre-processing (Yannic)

The Taskonomy ground truth is extracted from the Taskonomy GitHub repository [4] and stored in a Pandas DataFrame format[5]. We then analyzed the DataFrame by its unique tasks (20 Taskonomy tasks are included in the ground truth file) and excluded each task, be it a source or target task, if it was not one of the 17 Taskonomy tasks we used for comparison. We ignored in that way the following Taskonomy tasks and their TL results from the ground truth file:

- colorization
- class places
- inpainting whole

As a result, we obtained a DataFrame-structure as displayed in Table 6 that consists of information about single-source transfers to one target task or two-source task transfers to one target task and their respective TLP. We interpreted task combinations that use the same task name in source1 and source2 as a single-source transfer to one target and renamed the respective cell in the “source2" column to “na".

[4]https://github.com/StanfordVL/taskonomy .

[5]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Source1	Source2	Target	Ground_Truth
segment25d	na	rgb2sfnorm	0.01346
class_1000	na	edge3d	0.00430
...	...	...	...
curvature	reshade	segment25d	0.04501

TABLE 6: Example of what ground truth DataFrame looks like on Taskonomy with 471 transfers.

The ground-truth data for transfers to the Pascal VOC semantic segmentation task had already been prepared and stored in a list in the GitHub repository of DDS[6]. It consists of 17 source transfers to Pascal VOC semantic segmentation. Part of this list is displayed in Table 7.

Source	Target	Ground_Truth
autoencoder	Pascal VOC semantic segmentation	0.59016
class_1000	Pascal VOC semantic segmentation	0.64929
curvature	Pascal VOC semantic segmentation	0.65294
...	...	...
vanishing_point	Pascal VOC semantic segmentation	0.58918

TABLE 7: Ground truth on Pascal VOC with in total 17 transfers.

5.3.4 R^2 and Bayes Factor Calculation for Source and Target Task Combinations (Daniel)

With the creation the DataFrame as described in Section 5.3.3, we obtained a DataFrame in which each row represents one source/sources-target task combination with its TLP as the ground truth. In the following, we add two more columns to this DataFrame. These columns contain the prediction of the TLP from MOTS, one using R^2 and the other the BF. We use the following procedure to accomplish this: This happens in the following procedure:

1. We created the RDMs based on the different configuration cases from Section 5.3.2.
2. For creating predictions of TLP, we iterated over the rows of the created DataFrame (see Section 5.3.3) and read out the one or two source tasks and the target task in the row.
3. We loaded the RDMs for the source/sources and for the target task.
4. We extracted the upper triangular of the one or two source RDMs and flattened the matrix to a vector containing the values of the source RDMs.
5. We also extracted the upper triangular of the target RDM and flattened it to a target vector.
6. We perform a linear regression based on the RDM/RDMs of the source/sources for predicting the RDM of the target task. The source-task RDMs were used as the independent variable of the linear regression and the target task RDM was the dependent variable.

[6]https://github.com/cvai-repo/duality-diagram-similarity

7. We calculated the BF value by creating a random null-model of the BF approach (explained in Section 2.6) for comparison and used the BF as the quality of fit for the linear regression in one of the two added columns.

8. We stored the R^2 value as an alternative measure for the quality of fit in the second of the two added columns.

Table 8 shows how the DataFrame looks at this point.

source1	source2	target	ground_truth	bayes_factor	r_2
segment25d	na	rgb2sfnorm	0.01346	564212	0.5019
class_1000	na	edge3d	0.0043	148045	0.1671
...	...	...	...	...	...
denoise	segment2d	edge2d	0.0094	410042	0.3974

TABLE 8: Example of the calculated DataFrame.

5.3.5 Correlation Calculation Between MOTS and Ground Truth (Yannic)

The results presented in Section 5.3.4 provide the opportunity to calculate the correlation between the ground truth and R^2 or the BF. We calculated the correlation using the Spearman ranked correlation coefficient explained in Section 2.6. The calculation is immediately processed with a single pre-built function offered in the Pandas library [7], which allowed us to specify the column and method for the correlation calculation. The "corr" function automatically ranks cells in a descending order and does that for the ground-truth column and the R^2 and BF columns. It then calculates the correlation based on the rankings of the ground truth and BF or ground truth and R^2.
We performed the correlation calculation in two steps. We first grouped each target task by its name and stored each individual group in a dictionary with the respective target name as the key. We then performed the calculation based on each of our target tasks so that we obtained the task-based correlation between the ground truth and our metric. To obtain the overall correlation of all tasks, we used the mean of all task-based correlations summed together and divided by the number of tasks.

5.3.6 Search for Optimal Image Amount for RDMs (Yannic)

This section provides an overview of several experiments we executed based on different RDM train-test split sizes. We report results on the basis of our created DataFrame, our calculated BF and R^2 values from Section 5.3.4, the task-based groupings in Section 5.3.5 and the Spearman correlation between the ground truth and our metric, R^2 or BF. The correlation between the ground truth and R^2 and the correlation between the ground truth and BF are nearly identical and only differ after the seventh decimal place. We therefore omit the presentation of both results and show one result rounded to four decimal places. We analyzed the behaviour of MOTS based on the different RDM sizes we used for the prediction and looked for the RDM size that gives the overall best performance.

We present results for single-source transfers, multi-source transfers (only two source-transfers) and all-source transfers (one and two-source transfers) on the Taskonomy

[7] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

dataset, on the Pascal VOC dataset and for domain difference to investigate the common problem involved in transfers from one dataset (Taskonomy) to a completely different dataset (Pascal VOC).

Single-Source Transfers:
Each executed experiment on the Taskonomy dataset, both with an equal RDM train-test split size for train and test in Table 9 and with a constant test-set size of 500 images in Table 15, showed good results and a high mean correlation value between TLP and MOTS with a value of around 0.86. It is noticeable, that all correlation values show very similar and very close results to each other, which excludes the possible consideration of MOTS being dependable on a certain number of images (be it small or large). Nonetheless, (around 0.87) was obtained with a RDM train and test size of 2,400.
Table 11 shows results for target-task-based correlations between TLP and MOTS for a train-test split size of 2,400. It especially outlines class_1000 and denoising as target tasks with a clearly lower correlation than the other ones (below 0.80).
We found a different pattern, when we executed our experiments on the Pascal VOC dataset with the target task Pascal VOC semantic segmentation shown in Table 12. Especially for the case of domain equality, there is a pattern that tells us, that a higher size for our RDM train and RDM test supports a better performance of MOTS, with the best overall performance at 2,200 with around 0.81. The perfomance for the case of domain difference (using images from Taskonomy to predict Pascal VOC semantic segmentation) was in most cases average at best, with the best result at a size of 50 (0.6789). MOTS underpins the issue of using TL in different domain areas, due to the differently learned representations of source tasks.

	Train-Train RDMs	Train-Test RDMs
Size \ Metric	MOTS	MOTS
50	0.8647	0.8627
200	0.8698	0.8621
400	0.8715	0.8629
600	0.8699	0.8686
800	0.8674	0.8775
1,000	0.8643	0.8787
1,200	0.8674	0.8777
1,400	0.8684	0.8766
1,600	0.8708	0.8758
1,800	0.8718	0.8804
2,000	0.8719	0.8785
2,200	0.8741	0.8807
2,400	0.8743	0.8797

TABLE 9: Mean correlation between TLP and MOTS for single-source transfers on 17 Taskonomy tasks and the Taskonomy dataset with a RDM-train- and test-set sizes being is equal.

	Train-Test RDMs
Size \ Metric	MOTS
50	0.8747
200	0.8766
500	0.8753
1,000	0.8746
1,500	0.8751
2,000	0.8764
2,500	0.8771
3,000	0.8776
3,500	0.8770
4,000	0.8759
4,500	0.8752

TABLE 10: Mean correlation between TLP and MOTS for single-source transfers on 17 Taskonomy tasks and the Taskonomy dataset with a fixed test-set size of 500.

	Train-Train Growing with RDM Size 2,400
Task/ Metric	MOTS
Autoencoder	0.8390
Class 1000	0.7538
Curvature	0.8118
Denoising	0.7854
Edge 2D	0.9276
Edge 3D	0.9364
Keypoiont 2D	0.8258
Keypoint 3D	0.8188
Reshade	0.9294
Depth	0.9767
RGB2Mist	0.9732
Surface Normals	0.8872
Room Layout	0.9013
Segment 2.5D	0.8416
Segment 2D	0.9215
Semantic Segmentation	0.8513
Vanishing Point	0.8820

TABLE 11: Target-task-based correlation between TLP and MOTS based on all source transfers to that target for single-source transfers on 17 Taskonomy tasks.

	Train-Train RDM on Pascal VOC	Train-Test RDM on Pascal VOC	Train-Train RDM on Taskonomy	Train-Test RDM on Taskonomy
Size \ Metric	MOTS	MOTS	MOTS	MOTS
50	0.6373	0.6618	0.5686	0.6789
200	0.7059	0.7010	0.5711	0.5270
400	0.6985	0.7475	0.5245	0.5172
600	0.7696	0.7745	0.5172	0.5882
800	0.7475	0.7623	0.5123	0.6103
1,000	0.7843	0.7770	0.5319	0.5662
1,200	0.7794	0.7941	0.5735	0.5539
1,400	0.7500	0.7745	0.5711	0.5172
1,600	0.7672	0.8039	0.5637	0.5686
1,800	0.7623	0.7990	0.5637	0.5931
2,000	0.7819	0.7917	0.5735	0.5368
2,200	0.8137	0.7941	0.5735	0.5368
2,400	0.7990	0.7574	0.5637	0.5466

TABLE 12: Mean correlation between TLP and MOTS for 17 Taskonomy tasks and Pascal VOC semantic segmentation as target task with a RDM-train and test-set size being is equal.

	Train-Test on Pascal VOC	Train-Test on Taskonomy
Size \ Metric	MOTS	MOTS
50	0.7574	0.5858
200	0.7255	0.5539
500	0.7255	0.5196
1,000	0.7255	0.5270
1,500	0.7255	0.5270
2,000	0.7451	0.5270
2,500	0.7451	0.5270
3,000	0.7451	0.5270
3,500	0.7451	0.5270
4,000	0.7451	0.4951
4,500	0.7451	0.4951

TABLE 13: Task-based correlation between TLP and MOTS on 17 Taskonomy tasks and Pascal VOC semantic segmentation as target task with a fixed test-set size of 500.

Multi-Source Transfers:
This paragraph describes experiments of multi-source transfers including only two source-tasks transfers. We performed experiments on the Taskonomy dataset, both with with an equal RDM train-test split size as shown in Table 9 and a fixed RDM test-set size of 500 images as shown in Table 15. We obtained multi-source transfer results, that reveal a contrary picture to single-source transfer results with a correlation, that is at best average (below 0.5) in each case.
Analyzing these results on each target task shown in Table 11 manifests the average

correlations we obtained. There are some tasks, that display negative correlation (keypoint 2D, semantic segmentation, etc.) or even a perfect correlation (Class 1000).

	Train-Train	Train-Test
Size/ Metric	MOTS	MOTS
50	0.4220	0.3552
200	0.3928	0.4538
400	0.4496	0.4130
600	0.4298	0.4013
800	0.4112	0.4070
1,000	0.4090	0.4067
1,200	0.4217	0.4113
1,400	0.4112	0.4331
1,600	0.4108	0.4300
1,800	0.4153	0.4301
2,000	0.4089	0.4325
2,200	0.4264	0.4112
2,400	0.4183	0.4161

TABLE 14: Multi-source transfers of 17 Taskonomy tasks on one target task with the Taskonomy dataset and with a RDM train- and test-set sizes being as equal.

	Train-Test RDMs
Size/ Metric	MOTS
50	0.3893
200	0.4190
500	0.4001
1,000	0.4140
1,500	0.4061
2,000	0.4029
2,500	0.4029
3,000	0.4029
3,500	0.4029
4,000	0.4029
4,500	0.3995

TABLE 15: Multi-source transfers to one target task on 17 Taskonomy tasks on Taskonomy dataset and a test-set size fixed of 500 images.

	Train-Test Growing with RDM Size 1,800
Task/ Metric	MOTS
Autoencoder	0.7714
Class 1000	1.0000
Curvature	0.4424
Denoise	0.7714
Edge 2D	0.0286
Edge 3D	0.0061
Keypoint 2D	-0.3143
Keypoint 3D	0.4788
Reshade	0.0667
Depth	0.6727
RGB2Mist	0.7939
Surface Normals	0.8667
Room Layout	0.5273
Segment 2.5D	0.5879
Segment 2D	-0.2121
Semantic Segmentation	-0.2364
Vanishing Point	0.7697

TABLE 16: Target-task-based correlation between TLP and MOTS based on all source transfers to that target.

All-Source Transfers:
Table 17 shows the results of our experiments executed on the Taskonomy dataset with 17 Taskonomy tasks and equal RDM train- and RDM test-set sizes, Table 18 shows the same information for a fixed test-set size of 500 images. Similar to the single-source case, we obtained good results and high mean correlation values between TLP and MOTS with values ranging from 0.89 to 0.91. All correlation values appear to be very similar and provide close results to one another. The best result (0.9038) was obtained with a RDM train-test size of 1,800. Table 19 presents results for target-task-based correlations between TLP and MOTS for a train-test split size of 1,800. Each target represents a high correlation with a value of at least 0.80, but there are more with values ranging above 0.90. By comparing these results (especially 19 with multi-source transfers or single-source transfers it is clear to see, that all-source transfers achieve better results than the other two independently. We therefore note, that a mix between single and multiple sources should be included in TL.

	Train-train	Train-test
Size/ Metric	MOTS	MOTS
50	0.8946	0.8890
200	0.8973	0.8953
400	0.9020	0.8971
600	0.9011	0.8990
800	0.9001	0.9028
1,000	0.8991	0.9038
1,200	0.9002	0.9023
1,400	0.9011	0.9022
1,600	0.9008	0.9019
1,800	0.9015	0.9038
2,000	0.9012	0.9032
2,200	0.9026	0.9035
2,400	0.9023	0.9037

TABLE 17: All-source transfers of 17 Taskonomy tasks on one target task with the Taskonomy dataset and with a RDM train-test sizes being as equal.

	Train-Test RDMs
Size/ Metric	MOTS
50	0.8993
200	0.9023
500	0.9011
1,000	0.9013
1,500	0.9013
2,000	0.9018
2,500	0.9018
3,000	0.9021
3,500	0.9021
4,000	0.9014
4,500	0.9010

TABLE 18: All-source transfers to one target task on 17 Taskonomy tasks on Taskonomy dataset and a test-set size fixed at 500 images.

	RDM Train-Test Growing with RDM Size 1,800
Task/ Metric	MOTS
Autoencoder	0.9167
Class 1000	0.8450
Curvature	0.8302
Denoise	0.9029
Edge 2D	0.8886
Edge 3D	0.8940
Keypoint 2D	0.9029
Keypoint 3D	0.9119
Reshade	0.9252
Depth	0.9634
RGB2Mist	🟩 0.9799
Surface Normals	0.9491
Room Layout	0.9334
Segment 2.5D	0.8777
Segment 2D	0.8777
Semantic Segmentation	🟥 0.8196
Vanishing Point	0.9462

TABLE 19: Target-task-based correlation between TLP and MOTS based on all source transfers to that target.

5.3.7 MOTS Results for Single-Source Transfer Learning Prediction (Daniel)

After creating our DataFrame with the ground truth values as well as our predictions of R^2 and BF, we compared our results with already existing methods. Since existing state-of-the-art methods can only use one source task for the prediction, we decided, for the purpose of comparability to analyze MOTS when just one source task is used. In this way, MOTS is directly comparable. Therefore, we temporarily left out all rows with two sources included in the DataFrame (see the example visualization in Table 20).

source1	source2	target	ground_truth	bayes_factor	r_2
segment25d	na	rgb2sfnorm	0.0135	564212	0.5019
class_1000	na	edge3d	0.0043	148045	0.1671
...	...	...	...	...	...
keypoint3d	na	rgb2depth	0.0317	759271	0.6085

TABLE 20: Example of the calculated DataFrame. na stands for "not available" and means in this case that there is no source2, and therefore the row is just a single-source transfer.

Futhermore, we filtered the DataFrame for each target task so that we generated a DataFrame for each target task in which all the rows are rows containing transfers to the mentioned target task. After that, we calculated the Spearman's correlation between the columns of the ground truth of the Taskonomy TLP and the column of R^2 as our prediction for each target task. Calculating the average over all tasks, we came to a TLP prediction of 0.880, thereby showing outperformance over existing state-of-the-art methods. Table 63 shows the correlation of single-source TLP prediction with the ground truth on the Taskonomy dataset grouped by each target task.

Since the correlation of the ground truth and our metrics for the quality of fit for R^2 and the BF only differ at the seventh decimal point, we summarized both metrics to one column and named it MOTS.

Target Task	MOTS
autoencoder	0.8495
class_1000	0.7609
curvature	0.8118
denoise	0.8012
edge2d	0.8820
edge3d	0.9381
keypoint2d	0.8627
keypoint3d	0.8293
reshade	0.9416
rgb2depth	0.9785
rgb2mist	0.9838
rgb2sfnorm	0.8855
room_layout	0.9136
segment25d	0.8434
segment2d	0.9215
segmentsemantic	0.8670
vanishing_point	0.8960

TABLE 21: Correlation tasks of target task with ground truth using 1,800 Taskonomy images.

Pascal VOC:
After analyzing MOTS in a single-source context on the Taskonomy dataset, we as well evaluated it using the Pascal VOC dataset to prove that MOTS performance is generalizable over the Taskonomy dataset to other datasets as well. The procedure of this experiment was the same as described in this section for the Taskonomy experiment. However, as a target task, just the Pascal VOC target task was given, and for RDM creation, the images from the Pascal VOC dataset were used. Table 64 shows the results for MOTS with single-sources using the datasets Taskonomy and Pascal VOC and comparing them to the DDS method. It is clear that MOTS outperforms DDS with both cosine and Laplacian-function in the domain transfer case, where the Pascal VOC semantic segmentation task is predicted by using Taskonomy images for the RDMs (right column), however MOTS performs slightly worse (by 0.002) than DDS with Laplacian while predicting Pascal VOC semantic segmentation tasks while using Pascal VOC images for RDM creation.

Method	Pascal VOC	Taskonomy
DDS(f=cosine)	0.789	0.539
DDS(f=Laplacian)	**0.801**	0.581
MOTS	0.799	**0.593**

TABLE 22: **Correlation of single-source transfer learning for Pascal VOC semantic segmentation**. For both MOTS and DDS, the RDMs were computed with 1,800 images. We created RDMs using images of different datasets (Pascal VOC and Taskonomy).

5.3.8 MOTS Results for Multi-Source Transfer Learning Prediction (Daniel)

Based on our task-based RDMs and source-target combinations, we calculate the following:

- The Spearman correlation of the TLP and the MOTS prediction for each target task.
- The mean average Spearman correlation of the TLP and the MOTS prediction for all target tasks combined.

For this analysis, we included all rows of the DataFrame, that were multiple sources[8] (check column "source2" in Table 23). All rows with single-sources were dropped from the DataFrame. The results of the average correlation with the ground truth

source1	source2	target	ground_truth	bayes_factor	r_2
denoise	segment2d	edge2d	0.0094	410042	0.3974
rgb2sfnorm	reshade	edge3d	0.0658	1385668	0.8194
...	...	...	...	...	...
reshade	keypoint3d	rgb2depth	0.0521	1231554	0.7816

TABLE 23: Example of the calculated DataFrame.

for multiple sources are provided in Table 24. The average of these results regarding all target tasks is in 0.430. This result is much worse than that result from the single-source tasks (0.880) in Section 5.3.7. Looking in detail at the values in Table 24 reveals that the bad performance is attributable to the target tasks edge3d, keypoint2d, segment2d, segmentsemantic and reshade (marked in gray in Table 24).
To understand why these four tasks perform much worse, we performed a variance analysis of the Taskonomy affinity scores of the individual target tasks with two sources (provided in Table 25). By checking the variance of the tasks edge3d, keypoint2d, segment2d and segment semantic, it becomes clear that these five tasks with low MOTS prediction have a low variance of the Taskonomy TLP score. The conclusion we draw from this is that for the cases of target tasks in which the difference in TLP from different sources is not significant, the MOTS ranking prediction is not correct. However, since the difference in TLP from different sources is not significant, the TLP of the source proposed by MOTS will be close to the optimal TLP.

[8] In this analysis the maximum of sources for a transfer are two sources (see Table 23), since the provided ground truth of Transfer Learning Performance just contains single-source and double-source transfers

Target Task	MOTS
autoencoder	0.7714
class_1000	1.0000
curvature	0.4667
denoise	0.6571
edge2d	0.2571
edge3d	-0.2242
keypoint2d	-0.0286
keypoint3d	0.4788
reshade	0.1152
rgb2depth	0.6727
rgb2mist	0.7939
rgb2sfnorm	0.7697
room_layout	0.5273
segment25d	0.6242
segment2d	-0.0303
segmentsemantic	-0.3091
vanishing_point	0.7697

TABLE 24: MOTS correlation of multiple sources with RDMs created from 1,800 Taskonomy images with train-test and test set-growing.

target	Task Affinity(mean±var)	MOTS
rgb2sfnorm	0.0342±0.0053	0.7697
segmentsemantic	0.0316±0.0003	-0.3091
edge3d	0.0335±0.0007	-0.2242
rgb2depth	0.0322±0.0004	0.6727
keypoint2d	0.0371±0.0111	-0.0286
room_layout	0.0302±0.0002	0.5273
segment2d	0.0302±0.0002	-0.0303
autoencoder	0.0351±0.0071	0.7714
segment25d	0.0327±0.0003	0.6242
curvature	0.0319±0.0008	0.4667
edge2d	0.0388±0.0263	0.2571
vanishing_point	0.0319±0.0013	0.7697
keypoint3d	0.0335±0.0011	0.4788
rgb2mist	0.0323±0.0004	0.7939
class_1000	0.0243±0.0003	1.0000
denoise	0.0385±0.0304	0.6571
reshade	0.033±0.0005	0.1152

TABLE 25: **Per task mean and variance Taskonomy TLP score**
Transfer learning performance per target task compared to MOTS transfer learning correlation on Taskonomy dataset for 16 tasks.

5.3.9 MOTS Results for Single- and Multi-Source Transfer Learning Prediction (Daniel)

In the all-sources case, we combined the results from Sections 5.3.7 and 5.3.8, meaning we did not drop any rows in this case and worked with the full DataFrame in the form of Table 8 as described in Section 5.3.4. The average of the results of all target tasks was 0.904. The results for each target task are provided in Table 26.

Target Task	MOTS
autoencoder	0.9167
class_1000	0.8450
curvature	0.8302
denoise	0.9029
edge2d	0.8886
edge3d	0.8940
keypoint2d	0.9029
keypoint3d	0.9119
reshade	0.9252
rgb2depth	0.9634
rgb2mist	0.9799
rgb2sfnorm	0.9491
room_layout	0.9334
segment25d	0.8777
segment2d	0.8777
segmentsemantic	0.8196
vanishing_point	0.9462

TABLE 26: Correlations of single sources and multiple sources combined using 1,800 images for RDM creation from Taskonomy dataset with train-test and test-set growing.

5.4 MOTS for Multi-Task Architecture

5.4.1 Experimental Setup (Daniel)

One specific application for MOTS is the selection of source-task combinations from different task decoder heads in a multi-task architecture. Figure 29 shows a visualization of this application using the MTI-Net architecture explained in Section 2.3.4. The multi-task decoder heads of the multi-task architecture are treated like individual source-target combinations. For each decoder head, the features are extracted, and the RDMs are created out of the features for each head individually. The same is done for the target task, extracting the features from the final layer. Next, the linear regression of MOTS from the source tasks to the target tasks is performed for each possible combination of source tasks. This experiment was performed to evaluate MOTS in a new application to find the best source combination of task-heads from a multi-task architecture.

Ground Truth:
There is no ground truth provided for TL from different task-heads for the used multi-task learning architecture, MTI-Net (More information about why we chose MTI-Net for this analysis follows in Section 5.4.2). Therefore, the ground truth to compare MOTS against was generated in the process of this thesis. We apply different TN architectures for TL in Section 5.4.9 for generating the ground truth ourselves and evaluates the different TL approaches in Section 5.4.10.

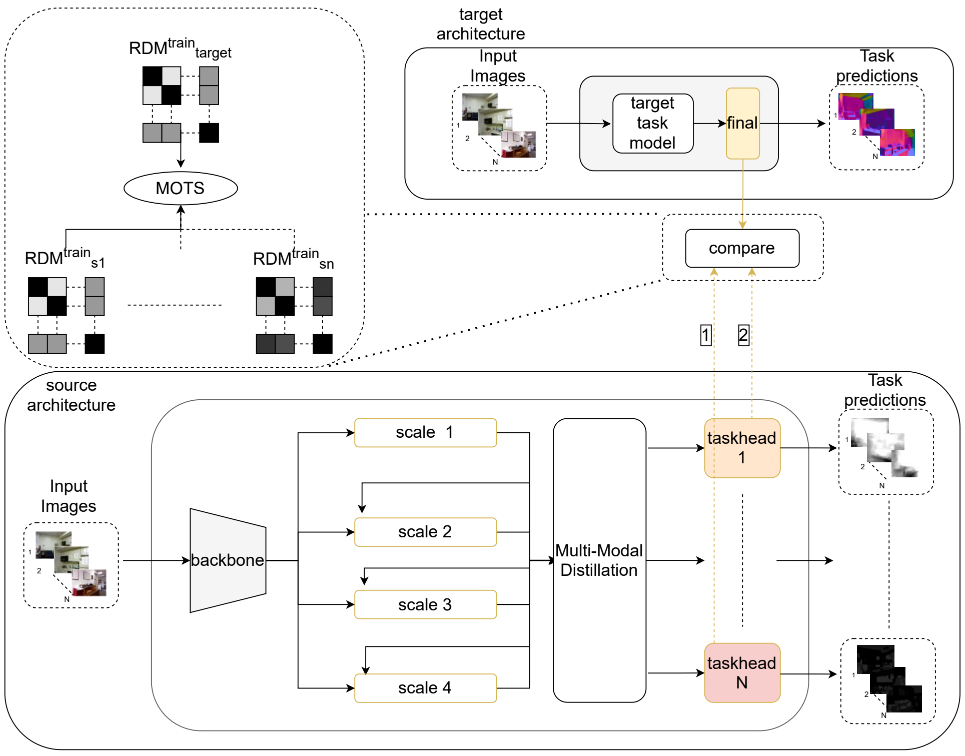


FIGURE 29: **Estimating task similarity between a target task and multi-task decoders.** The multi-task source-target combination consists of multiple task heads each corresponding to a computer vision task. In this work, individual task heads (or combinations of them) from a multi-task source-target combination are used as sources to transfer to a target task. From each task head, we first extract the feature maps and then create RDMs. Similarly, the feature map is extracted from the final layer of the target architecture to compute the target RDM. To compute the task similarity between source task heads and target task representations, we use MOTS, which predicts the target task's RDM linearly from source RDMs and evaluate the fit quality with R^2.

Tasks:
For NYUD this analysis uses the tasks depth (dp) estimation, surface normals (sn) and semantic segmentation (ss). For Pascal VOC, the analysis uses the tasks human parts (hp), saliency (sl), ss and sn. The tasks are explained in Table 27. We trained three MTI-Net models for NYUD and four MTI-Net models on Pascal VOC. For each training, we selected one of the tasks as the target task and set the others (two for NYUD and three for Pascal VOC) as the source tasks. The source-target combinations use the following notation:

- NYUD: $source1\_source2 = target$
- Pascal VOC: $source1\_source2\_source3 = target$.

The number of possible sources for TL is the equivalent of the powerset of the set of source tasks from the multi-task architecture. Table 28 shows the source-target combinations used with the NYUD dataset, while Table 29 displays the source-target combinations used with the Pascal VOC dataset.
While training, the target tasks were evaluated on different metrics. Table 30 shows the MTI-Net reported metrics for each target task. In this work, we selected only one metric per target task. For sn, ss and dp we select root mean squared error (RMSE)

Task	Explanation
depth	For each pixel in the image, a depth value is estimated [26], [68].
surface normals	In surface normal estimation, the physical orientations of the pixels from the image are predicted [79].
semantic segmentation	Semantic segmentation is a combination of identifying multiple objects in an image and classify the corresponding pixels of the objects with the object class. Therefore, in semantic segmentation each pixel is assigned a class label [44] [70].
human parts	Human parts are segmented on the image, meaning that human parts are identified on the image, and each pixel color is based on the class of the part identified [33].
saliency	In saliency estimation, the borders between objects and their neighbors are amplified. With that, saliency estimation assigns different levels of visual relevance to distinct regions of the image [7].

TABLE 27: Used tasks with explanation.

Source-Target Combinations for NYUD			
Source 1	Source 2	Target	Abbreviation
depth	surface normals	semantic segmentation	dp_sn=ss
surface normals	semantic segmentation	depth	sn_ss=dp
depth	semantic segmentation	surface normals	dp_ss=sn

TABLE 28: NYUD task combinations for multi-task experiment.

while for hp and sl we select mIoU.

Root mean squared error:
The RMSE is a typically used metric for evaluating the prediction error of a model. It is the standard deviation of the prediction errors (also known as residuals). The formula of RMSE is as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y_i} - y_i)^2}{n}} \tag{5.2}$$

where $\hat{y_1}, ..., \hat{y_n}$ are the predicted values, $y_1, ..., y_n$ are the observed values and n is the number of observations.

mean Intersetion over Union:
mIoU is a widely used evaluation metric, for example for semantic image segmentation tasks [67]. The calculation of mIoU proceeds as follows:

1. Calculate the Intersection over Union (IoU) for each class using the following formula:

$$IoU = \frac{tp}{tp + fp + fn} \tag{5.3}$$

Source-Target Combinations for Pascal VOC				
Source 1	Source 2	Source 3	Target	Abbreviation
human parts	saliency	semantic segmentation	surface normals	hp_sl_ss=sn
human parts	surface normals	saliency	semantic segmentation	hp_sn_sl=ss
surface normals	saliency	semantic segmentation	human parts	sn_sl_ss=hp
human parts	surface normals	semantic segmentation	saliency	hp_sn_ss=sl

TABLE 29: Pascal VOC task combinations for multi-task experiment.

Target Task	Metric
surface normals (sn)	mean, median, **RMSE**, 11.25, 22.5, 30
semantic segmentation (ss)	**mIoU**
depth (dp)	**RMSE**, log-RMSE
human parts (hp)	**mIoU**
saliency (sl)	**mIoU**, maxF

TABLE 30: Target tasks and their metrics. The chosen metric is displayed in bold.

where tp are the true positive predicted pixels, fp are the false positive predicted pixels and fn are the false negative predicted pixels

2. Calculate the mean of all IoU scores.

The Intersection over Union is also known as the Jaccard-distance [71].

5.4.2 Used Architecture (Yannic)

The choice of a suitable CV architecture for the MOTS experiment is not so trivial as it might seem, especially for the case of multi-source transfers. ST models at hand are not capable of producing more than one output. Using the source heads of many different ST models together for the transfer to one target task would be computationally costly. ST models were therefore excluded as a potential source-model solution. Thus, a suitable architecture needed to be selected from a set of different MTL models that promise good results and efficiency over a ST model due to the processing of multiple tasks in just one model.

Different types of MTL architectures were presented in Section 2.3, and they are all suitable as a source model. Nonetheless, we decided to use the MTI-Net due to its efficiency, innovative approach with its initial task predictions, inclusion of auxiliary tasks, processing of the tasks at different scales as well as because of the achieved results compared to other models [73].

We used the following architectures in combination:

- Backbone network HRNet
- MTI-Net

- Transfer net (TN), described later in 5.4.9

The process flow is visualized in Figure 30. HRNet predicts and provides initial task-specific heads at different scales. MTI-Net processes these initial task heads in combination and outputs task-specific results at the end. We then used an additional network defined by us as a TN architecture, that combines source-specific heads to produce the final target task.

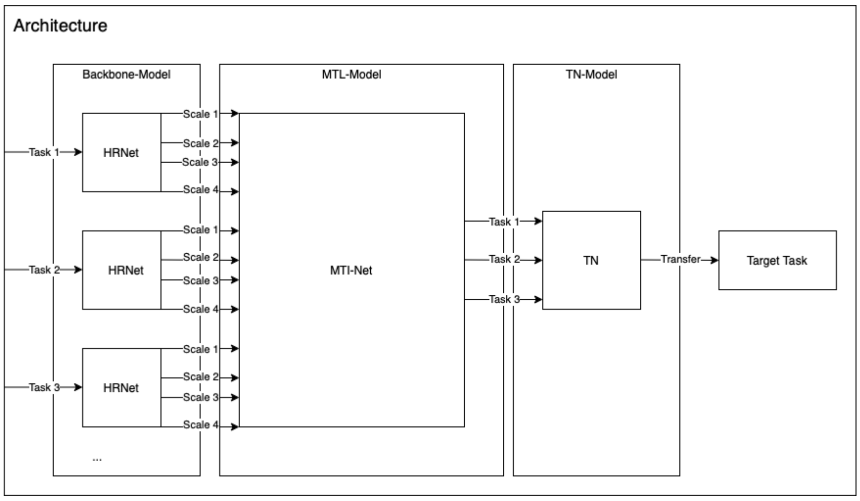


FIGURE 30: Architecture, we used for the experiment.

5.4.3 Single-Task Baseline (Yannic)

We use as a baseline model a ST architecture, which only accepts and outputs one specific task. The ST model is structured similar to the MTI-Net but without layers, that are used when processing more than one tasks and as well without multiple-task decoder heads at the end. MTI-Net is built on this model in order to be comparable with it performance wise. The training process of the MTI-Net consists of the following two points:

- Train ST baselines on one specific task described in Table 27.
- Start MTI-Net training in different task-combinations shown in Table 28 for NYUD and in Table 29 for Pascal VOC.

5.4.4 Training Process (Yannic)

The training process is split into three different steps. First, the ST source-target combination is trained on one specific target task. Second, the MTI-Net source-target combination is trained through a combination of different source tasks to predict multiple outputs. Third, these predicted target tasks are then used as source tasks to predict one specific target task by performing TL on them. More regarding this is discussed in the Subsection feature extraction 5.4.6 and TN 5.4.9.
The configuration of the baseline source-target combination can be seen in Table 31 for the datasets NYUD and in Table 32 for Pascal VOC. The configuration of the MTI-Net source-target combination is visualized in Table 33 with the respective loss weights for Pascal VOC in Table 35 and for NYUD in Table 34.

In order to perform TL from a multi-source task perspective, we first needed to train our MTI-Net source-target combination and evaluate the results. MTI-Net outputs different target task predictions that can then be used for TL. A new target task is predicted through TL and feature extraction of the task-specific heads, that were generated previously.

	NYUD		
Configuration	ss	dp	sn
trBatch	8		
valBatch	1		
nworkers	4		
epochs	100		
optimizer	adam		
optimizer-lr	0.0001		
optimizer-weight-decay	0.0001		
scheduler	poly		
model	baseline		
backbone	HRNet-w18		
backbone-pretrained	True		
backbone-dilated	False		
head	HRNet		

TABLE 31: Single-task source-target model configuration training process for NYUD.

	Pascal VOC			
Configuration	ss	sn	sl	hp
trBatch	8			
valBatch	1			
nworkers	4			
epochs	60			
optimizer	sgd			
optimizer-lr	0.01*			
optimizer-weight-decay	0.0001			
scheduler	poly			
model	baseline			
backbone	HRNet-w18			
backbone-pretrained	True			
backbone-dilated	False			
head	HRNet			

TABLE 32: Single-task source-target combination configuration training process for Pascal VOC (*0.0001 for single-source transfer to ss).

Configuration	NYUD	Pascal VOC
trBatch	8	
valBatch	8	
nworkers	4	
epochs	100	
optimizer	adam	
optimizer-lr	0.0001	
optimizer-weight-decay	0.0001	
scheduler	poly	
model	MTI-Net	
backbone	HRNet-w18	
backbone-pretrained	True	
backbone-dilated	False	
head	HRNet	

TABLE 33: MTI-Net-configuration for training on datasets NYUD and Pascal VOC.

NYUD	
Tasks	Loss Weights
semseg	1.0
surface normals	10.0
depth	1.0

TABLE 34: Loss weights for MTI-Net model on NYUD.

Pascal VOC	
Tasks	Loss Weights
semseg	1.0
human parts	2.0
saliency	5
surface normals	10.0

TABLE 35: Loss weights for MTI-Net model on Pascal VOC.

5.4.5 Specifying Auxiliary Tasks for Training (Daniel)

Since the implementation of MTI-Net allows for a specification of source, target and auxiliary tasks, the question arises as to how the auxiliary tasks should be specified. It is necessary to declare that a task used as a source task also needs to be specified as an auxiliary task (forced by MTI-Net implementation); however, this does not hold in reverse. The question arises as to whether, all four auxiliary tasks should be used or if just the auxiliary tasks that are also used as source tasks should be used. Table 36 shows the results of the training process of MTI-Net with:

1. all auxiliary tasks used
2. equal sets of auxiliary tasks and source tasks used

Since the results are on average slightly better in the case of equally specified auxiliary tasks, the rest of the analysis only uses experiments with equal auxiliary tasks configuration.

Source-Target Combination	Source Task	All Auxiliary Tasks	Equal Auxiliary Tasks
sn_sl_hp=ss	hp	61.5703	61.2521
	sl	67.164	66.991
	sn	14.7886	14.5939
ss_sl_hp=sn	ss	61.9234	63.5639
	hp	61.8103	62.4998
	sl	67.348	67.498
ss_sn_hp=sl	ss	61.9955	62.1316
	hp	61.6819	62.0944
	sn	14.8787	14.8212
ss_sn_sl=hp	ss	62.0474	60.4335
	sl	67.152	67.123
	sn	14.8085	14.5673

TABLE 36: Comparison of MTI-Net training results for all auxiliary tasks and equal auxiliary tasks. The better values are colored in green. The source-target combinations sn_sl_hp=ss, ss_sn_hp=sl, ss_sn_sl=hp are given in mIoU (higher is better), while ss_sl_hp=sn is measured by the mean error (lower is better).

5.4.6 Feature Extraction (Daniel)

After training the single-task baseline and the MTI-Net models with different source and target combinations, it is necessary to create feature maps for all source-target combinations. These feature maps are used later to predict the TLP.
The feature maps are created by setting the batch size for validation to one and saving the representations within the MTI-Net model for each forward pass and therefore for each input image. Since it is not trivial which feature map size should yield better results, this analysis uses two different sizes. The two evaluated pixel sizes are 32x24px and 64x48px which we label as small and normal, respectively (see Figure 37).
Due to computational constrains, small feature extractions were preferred in the analysis. To keep the computational costs as low as possible without losing generalizability, a correlation analysis was performed between the feature maps of small and normal for one source-target combination: the feature maps of the task combination with depth and surface normals as source tasks and semantic segmentation as the target task using NYUD dataset. We hypothesized that the correlation for RDMs from the Pascal VOC dataset would behave similarly. For the correlation analysis, we first created the RDMs for depth and surface normals using the small size once and the normal size once. Then, we correlated both sizes for each source task by using the upper triangular of the RDMs and calculating the Pearson correlation of the upper triangles. We used the final layer of both tasks for the extraction of the feature maps for this correlation analysis (More information about the selection of the final layer is provided in Section 5.4.8). For both tasks, we achieved a correlation between small and normal that was higher than 0.9999 (value is rounded). This led us to the conclusion that we could use the small size for all further experiments without important information loss.

Label	Width	Height
small	32	24
normal	64	48

TABLE 37: Pixel sizes for the extracted features for small and normal sizes.

5.4.7 RDM Creation (Daniel)

For creating the RDMs, the features from Section 5.4.6 were used. These RDMs are based on 250 images in the NYUD case and 500 images in the Pascal VOC case. Using these RDMs, we performed ordinary least squares regression from the MOTS method. The processes of feature extraction, RDM creation and MOTS prediction were performed for multiple layers in the MTI-Net.

5.4.8 Layer Selection (Daniel)

In addition to the question of which size to use for the feature extraction, the non-trivial question of from which part of the network architecture the feature maps should be extracted needs to be answered.
For this, we identified five different positions in the MTI-Net architecture from which the feature maps are extracted (see Figure 31 for a visualized overview of the locations). Four of the positions for extraction are based on the four different scale levels used in the MTI-Net architecture: 1/32, 1/16, 1/8 and 1/4. In addition to these four locations, we extracted the feature maps from the task-specific heads (we call this final) directly before predicting the task at the rectified linear unit (ReLU)-layer. The ReLU-layer is a layer of neurons in a neural network which uses the rectified linear activation function:

$$ReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

where x is the input from the former layer. The ReLU function is a broadly used activation function in neural networks [8] and it outputs the results of the prediction for the target task in this case.

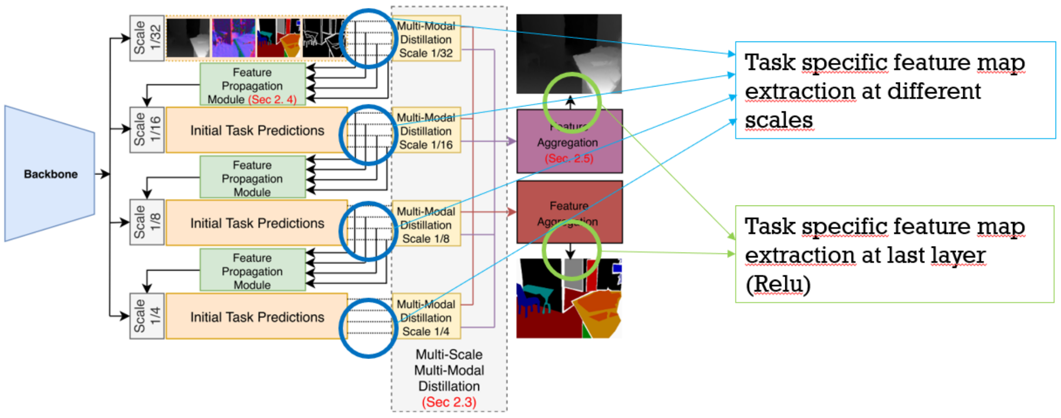

Backbone
Scale 1/32
Scale 1/16
Scale 1/8
Scale 1/4
Feature Propagation Module (Sec 2. 4)
Feature Propagation Module
Feature Propagation Module
Initial Task Predictions
Initial Task Predictions
Initial Task Predictions
Multi-Modal Distillation Scale 1/32
Multi-Modal Distillation Scale 1/16
Multi-Modal Distillation Scale 1/8
Multi-Modal Distillation Scale 1/4
Multi-Scale Multi-Modal Distillation (Sec 2.3)
Feature Aggregation (Sec. 2.5)
Feature Aggregation
Task specific feature map extraction at different scales
Task specific feature map extraction at last layer (Relu)

FIGURE 31: The five locations for feature extraction.

To compare the different locations of extracted features, we generated RDMs for the features of all locations. Afterwards, we used these RDMs to apply our MOTS

method and create the TLP prediction for all sources to the target task for each source-target combination.
This was performed for all three source-target combinations of NYUD and all four source-target combinations of Pascal VOC in order to determine the layer for performing TL on later on.
We plotted the results in one table per source-target combination. Tables 38, 39 and 40 display the quality of fit in respect to the predictions based on the NYUD dataset, while Tables 43, 42, 44 and 41 represent the predictions based on Pascal VOC. All values represent the R^2 value for the specific source-target combination. The column called "layer" displays the layer from which the representations are drawn. The variable f refers to the final layer, while $s1$, $s2$, $s3$, $s4$ refer to scale1, scale2, scale3 and scale4, respectively, of the MTI-Net architecture (see Figure 31). The highest values are marked in green.
For the sources-target tasks combinations dp_sn=ss (NYUD), dp_ss=sn (NYUD), sn_ss=dp (NYUD), hp_sn_sl=ss (Pascal VOC) and hp_sn_ss=sl (Pascal VOC) the final layer is clearly the best. For the source-target task combination of sn_sl_ss=hp (Pascal VOC), layer scale4 yields the best result, and for hp_sl_ss=sn (Pascal VOC), there is no clear best result. However, since all the predictions of sn_sl_ss=hp (Pascal VOC) and hp_sl_ss=sn (Pascal VOC) are close to zero, we argue that the best choice is to use the final layer for all the source-target task combinations for consistency. This suggests that for the given multi-task model (MTI-Net), the best choice for TL is to take the last layer of all the task heads to perform the transfer in all the combinations tested.

Layer	dp	sn	dp_sn
f	0.180060	0.173983	0.199403
s1	0.072420	0.106345	0.122618
s2	0.072111	0.097630	0.115699
s3	0.072560	0.105867	0.123954
s4	0.071813	0.118327	0.135859

TABLE 38: NYUD with source-target combination dp_sn=ss and the usage of train_test RDMs.

Layer	dp	ss	dp_ss
f	0.355330	0.186888	0.387210
s1	0.137628	0.185873	0.260321
s2	0.138012	0.198627	0.268709
s3	0.138585	0.204601	0.273481
s4	0.136019	0.187343	0.263634

TABLE 39: NYUD with source-target combination dp_ss=sn and the usage of train_test RDMs.

Layer	sn	ss	sn_ss
f	0.310605	0.187587	0.350263
s1	0.174823	0.128428	0.242082
s2	0.168223	0.191088	0.275776
s3	0.175603	0.190402	0.274621
s4	0.197111	0.176518	0.277418

TABLE 40: NYUD with source-target combination sn_ss=dp and the usage of train_test RDMs.

Layer	sn	sl	ss	sn_sl	sn_ss	sl_ss	sn_sl_ss
f	-0.000050	-0.000031	-0.000027	-0.000055	-0.000068	-0.000035	-0.000085
s1	-0.000032	-0.000042	-0.000058	-0.000029	-0.000052	-0.000053	-0.000066
s2	-0.000051	-0.000045	-0.000048	-0.000027	-0.000020	-0.000032	-0.000064
s3	-0.000005	-0.000047	-0.000008	-0.000047	-0.000015	-0.000048	-0.000066
s4	-0.000017	-0.000008	0.000002	-0.000018	-0.000034	-0.000007	-0.000036

TABLE 41: Pascal VOC with source-target combination sn_sl_ss=hp and the usage of train_test RDMs.

Layer	hp	sn	sl	hp_sn	hp_sl	sn_sl	hp_sn_sl
f	0.214835	0.180209	0.126714	0.263397	0.253411	0.236178	0.290947
s1	0.106013	0.078255	0.060696	0.151516	0.142811	0.118084	0.177888
s2	0.098511	0.072963	0.062718	0.141089	0.139134	0.116740	0.171996
s3	0.097672	0.068222	0.068951	0.137849	0.145548	0.118974	0.175886
s4	0.128403	0.062165	0.073412	0.160150	0.167154	0.119107	0.192697

TABLE 42: Pascal VOC with source-target combination hp_sn_sl=ss and the usage of train_test RDMs.

Layer	hp	sl	ss	hp_sl	hp_ss	sl_ss	hp_sl_ss
f	-0.000012	-0.000043	-0.000035	-0.000059	-0.000038	-0.000042	-4.801e-05
s1	-0.000012	-0.000005	-0.000013	-0.000020	-0.000014	-0.000033	6.876e-07
s2	-0.000028	-0.000003	-0.000014	-0.000003	-0.000024	-0.000011	-1.212e-05
s3	-0.000043	-0.000007	-0.000036	-0.000028	-0.000026	-0.000047	-3.132e-05
s4	-0.000027	-0.000004	-0.000012	0.000002	-0.000013	-0.000023	-8.670e-06

TABLE 43: Pascal VOC with source-target combination hp_sl_ss=sn and the usage of train_test RDMs.

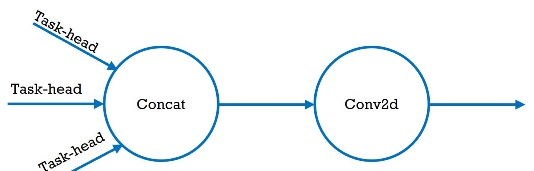
Layer	hp	sn	ss	hp_sn	hp_ss	sn_ss	hp_sn_ss
f	0.183031	0.117546	0.207511	0.210219	0.231044	0.235385	0.248366
s1	0.044525	0.054643	0.090897	0.081781	0.102432	0.124062	0.130192
s2	0.036935	0.058882	0.094606	0.079071	0.102156	0.128898	0.132250
s3	0.028917	0.056827	0.141222	0.071794	0.143372	0.165298	0.165879
s4	0.067396	0.053210	0.181539	0.099164	0.194024	0.197240	0.206244

TABLE 44: Pascal VOC with source-target combination hp_sn_ss=sl and the usage of train_test RDMs.

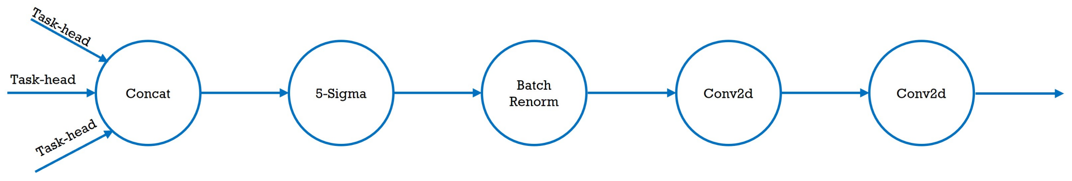
5.4.9 Transfer Nets (Yannic)

A TN was added to the best performing layer. We used a set of different TNs in order to determine, which one performs the best. For the implementation of the TN, we followed the approach of the Taskonomy paper, which is described in the supplementary material of Taskonomy in [84] and found in the GitHub repository [25]. In

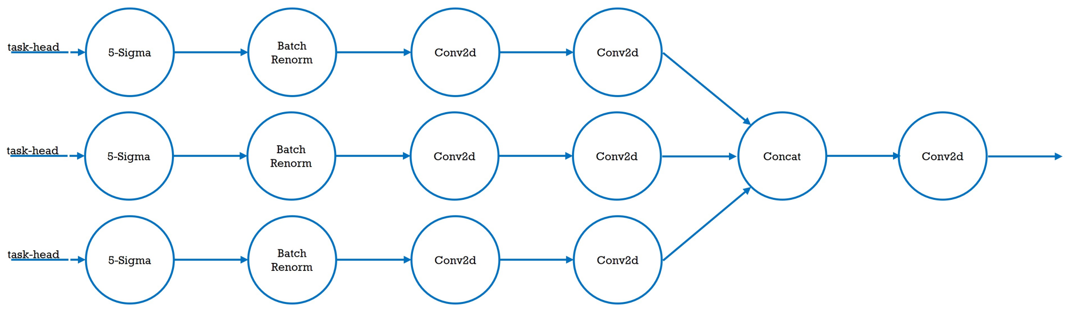
[84], a tower structure is used that consists of four different steps. First, extreme values are clipped to 5-sigma;[9] batch renormalization is processed afterwards, followed by two convolution layers at the end. In total, four different TNs were implemented and compared with one another in our experiment. They are visualized in Figure 32.



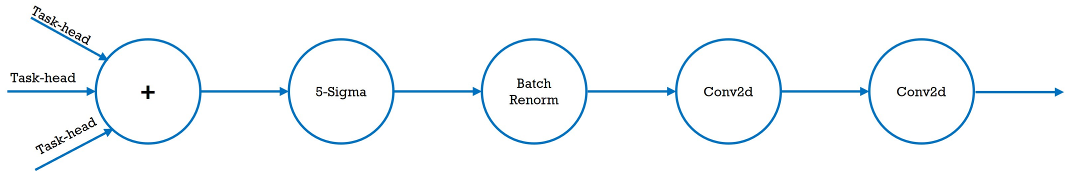
(a) Transfer Net 1



(b) Transfer Net 2



(c) Transfer Net 3



(d) Transfer Net 4

FIGURE 32: Four different transfer nets that are used as transfer learning layers for the prediction of one target task based on a combination of different source-tasks.

[9]Derived from the normal distribution with five-times sigma and therefore the following formula: $f(x) = \frac{1}{5\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{5\sigma})^2}$

Transfer Net 1				
Configuration	In Channels	Out Channels	Kernel Size	Stride
Conv2D	n-tasks * n-features	target output size	(1,1)	(1,1)

TABLE 45: Transfer Net 1 configuration.

Transfer Net 2					
Configuration	In Channels	Out Channels	Kernel Size	Stride	Padding
BatchRenorm2D	n-tasks * n-features				
Conv2D	n-tasks * n-features	n-features	(3,3)	(1,1)	(1,1)
Conv2D	n-features	target output size	(3,3)	(1,1)	(1,1)

TABLE 46: Transfer Net 2 configuration.

Transfer Net 3					
Configuration	In Channels	Out Channels	Kernel Size	Stride	Padding
BatchRenorm2D	n-features				
Conv2D	n-features	0.5*n-features	(3,3)	(1,1)	(1,1)
Conv2D	0.5*n-features	0.5*0.5 n-features	(3,3)	(1,1)	(1,1)
Conv2D	n-tasks * 0.5*0.5*n-features	target output size	(1,1)	(1,1)	

TABLE 47: Transfer Net 3 configuration.

Transfer Net 4					
Configuration	In Channels	Out Channels	Kernel Size	Stride	Padding
BatchRenorm2D	n-features				
Conv2D	n-features	n-features	(3,3)	(1,1)	(1,1)
Conv2D	n-features	target output size	(3,3)	(1,1)	(1,1)

TABLE 48: Transfer Net 4 configuration.

TN 1 is used as a computational resource-saving alternative, with just one convolution layer. Each task-specific head is first concatenated with the others and then processed by one convolution layer with the output dimension of the specific target task. TNs 2 and 3 follow the same structure as the Taskonomy approach by applying a tower structure to their TN [84]. In TN 2, each task-specific head is first concatenated with the others before being processed by the tower structure, whereas in TN 3, the tower structure is being used already at each task-specific head. A concatenation occurs afterwards, which is then accompanied by a further convolution layer and an output dimension with respect to the target task. TN 4 is handled the same

as TN 2, but with the element-wise addition instead of concatenation.
Each TN clips extreme values to 5-sigma[72]. Batch renormalization was used with a self-made function that is offered in GitHub [24]. Each TN configuration is described in more detail in Tables 45, 46, 47 and 48.

5.4.10 Selection of Optimal Transfer Net for Ground-Truth Creation (Daniel)

We experimented with the four different TN architectures described in Section 5.4.9. For performing the TL, we used the concepts of feature extraction and finetuning.
In feature extraction (fe), the layer parameters from the source tasks are copied and used identically in the target task. Just one or a few last layers were trained while training the model on the target task [69]. All other layer weights were not changed during the process since they were frozen.
The contrarian concept is that of fine tuning (ft), in which the layer parameters are also copied from the source task and transferred to the target task. However, while training the model on the target task, the whole network is trained. With this approach, the model has many more parameters that can be altered to fit during the training process. [69]
During the selection of the optimal TN both concepts are applied. For the fe approach the process is the following:

1. Freeze all layer parameters of the architecture.
2. Remove the output layer.
3. Add a TN from Section 5.4.9.
4. Train the architecture by training only the layers of the TN.

The ft process is nearly the same. The difference is that the layers are not frozen, and therefore the whole architecture is trained after adding the TN.
Tables 49, 50 and 51 display the results for the source-target combinations using the NYUD dataset, while Tables 52, 53, 54 and 55 provide the results of the transfers using the source-target combinations of the Pascal VOC dataset. The best results are marked in green, while the second best are marked in yellow. In Tables 50, 51, 52, 54 and 55, the results of TN3 are missing. The reason is that the transfer could not be computed with our hardware since the GPU RAM storage is insufficient.
In Figure 32, it is clear that TN1 is the least expensive in terms of complexity and therefore computational intensity, while TN3 is the most expensive. The reason is that TN1 has just one additional layer in comparison to the provided default MTI-Net architecture since we deleted the last layer of the default MTI-Net architecture and added the two layers of TN1, while TN3 has 13 more layers. This means that for training TN3, many more parameters need to be trained, which increases the training time as well as the memory footprint due to more saved parameters. Therefore, it fails to train TN3 due to insufficient GPU RAM on NVIDIA Tesla V100 SXM2 with 32GB of RAM on g5-server. At this point in the analysis, the g4-server was not available to train. It is unclear whether the 40GB of GPU RAM in the g4-server would have been sufficient.
Looking at the results for fe in isolation, we came to the conclusion that in most cases, TN2 provides the best results. Only TN4 provides better results in some cases, while this occurs mainly in Table 55 with the target task saliency.
Due to the computational constraints of this thesis, we only evaluated TN1 with the approach of ft. TN2, TN3 and TN4 are not computable with the used hardware

due to out-of-memory errors from the GPU. They are not even performable on the stronger g4-server. However, the ft results on TN1 show that it outperformed the results from fe by far for all given TNs. We hypothesize that the stronger performance of ft comes from the larger number of trainable parameters in ft compared to fe. With more parameters, the model has more possibilities to adopt to the training data. Similarly, the cause of the GPU out-of-memory error seems to be the greater memory usage due to more trainable parameters for ft than for fe.

dp_sn=ss					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
dp	0.0628	0.1367	0.1272	0.1366	0.1801
sn	0.0647	0.1307	0.1222	0.1312	0.1740
dp_sn	0.0874	0.1677	0.1555	0.1437	0.1994

TABLE 49: Feature extraction (fe) and finetuning (ft) results for dp_sn=ss with metric mIoU (highest is best).

sn_ss=dp					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
sn	1.2433	1.0872	-	1.0627	0.3106
ss	1.2784	1.0971	-	1.1326	0.1876
sn_ss	1.1554	1.0134	-	1.0844	0.3503

TABLE 50: Feature extraction (fe) and finetuning (ft) results for sn_ss=dp with metric RMSE (lowest is best).

dp_ss=sn					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
dp	44.4314	37.7339	-	37.7881	0.3548
ss	45.2671	43.0672	-	43.0753	0.1869
dp_ss	43.6890	37.0614	-	40.4617	0.3872

TABLE 51: Feature extraction (fe) and finetuning (ft) results for dp_ss=sn with metric RMSE (lowest is best).

hp_sl_ss=sn					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
hp	29.0942	28.5435	-	28.5202	18.5929
sl	28.6353	28.2802	-	28.3163	18.5907
ss	29.5513	28.7688	-	28.7723	18.5120
hp_sl	27.9470	27.4432	-	28.1451	18.6192
hp_ss	28.2681	27.6783	-	28.4109	18.6262
sl_ss	28.2211	27.6675	-	28.4530	18.5907
hp_sl_ss	27.5905	26.9392	-	28.2778	18.6030

TABLE 52: Feature extraction (fe) and finetuning (ft) results for hp_sl_ss=sn with metric RMSE (lowest is best).

hp_sn_sl=ss					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
hp	0.2001	0.2191	0.2161	0.2177	0.6333
sn	0.1416	0.1992	0.198	0.2001	0.6295
sl	0.0820	0.1134	0.1268	0.1004	0.6270
hp_sn	0.2340	0.2702	0.2665	0.2404	0.6321
hp_sl	0.2121	0.2383	0.2295	0.2208	0.6338
sn_sl	0.1755	0.2294	0.2270	0.2068	0.4861
hp_sn_sl	0.2439	0.2790	0.2790	0.2364	0.5397

TABLE 53: Feature extraction (fe) and finetuning (ft) results for hp_sn_sl=ss with metric mIoU (highest is best).

sn_sl_ss=hp					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
sn	0.2712	0.3401	-	0.3436	0.5984
sl	0.2217	0.3139	-	0.3139	0.6020
ss	0.3689	0.3965	-	0.4010	0.6021
sn_sl	0.3096	0.3890	-	0.3610	0.6004
sn_ss	0.3975	0.4299	-	0.4159	0.6008
sl_ss	0.3800	0.4163	-	0.3800	0.6013
sn_sl_ss	0.4039	0.4340	-	0.4124	0.6002

TABLE 54: Feature extraction (fe) and finetuning (ft) results for sn_sl_ss=hp with metric mIoU (highest is best).

hp_sn_ss=sl					
	fe				ft
Source	TN1	TN2	TN3	TN4	TN1
hp	0.0542	0.0566	-	0.0618	0.0362
sn	0.0659	0.1035	-	0.1008	0.0341
ss	0.0442	0.0517	-	0.0566	0.0301
hp_sn	0.0542	0.0721	-	0.0882	0.0300
hp_ss	0.0450	0.0434	-	0.0531	0.0336
sn_ss	0.0472	0.0660	-	0.06046	0.0299
hp_sn_ss	0.0482	0.0526	-	0.0626	0.0362

TABLE 55: Feature extraction (fe) and finetuning (ft) results for p_sn_ss=sl with metric mIoU (highest is best).

5.4.11 Comparison of MOTS Prediction and Ground-Truth Ranking (Daniel)

As described in Section 5.4.10, we trained four different versions of TNs on the task-decoder heads of the MTI-Net and then compared the results to the ranking of MOTS prediction method. This subsection displays the results of the four TNs and compares them with the MOTS prediction as well as with TL results created by using the single-task baseline models provided by the MTI-Net-framework. For the comparison of the MOTS prediction with the generated ground truth, we selected the TN1 ground truth for the ft approach and the TN2 ground truth for feature extraction due to it having the highest TLPs. The single-source transfers using the baseline single-source models provided by the MTI-Net framework were trained using TN2 with the fe-approach.
Tables 56, 57 and 58 display the results for the NYUD dataset, while Tables 59, 60, 61 and 62 display the results for the Pascal VOC dataset. All tables include the prediction performance given by the ground truth in the metric mIoU or RMSE. The column on the right of each table shows the values predicted by our MOTS TL prediction. It is necessary to highlight that the rows of the MOTS prediction of each source-target combination cannot be compared by their absolute values. However, the ranking of the rows is important. The similarity of the ranking from the MOTS prediction with the ranking of the ground truth displays the quality of the prediction method: the higher the similarity, the better. For a perfect prediction method, the ordering should be equal in all cases. The order of the three best sources for use in TL is displayed for the first-, second- and third-best options in green, yellow and red, respectively.

Ground Truth (sn_ss=dp)			Prediction
Metric / Source	RMSE (fe)	RMSE (ft)	MOTS
sn	1.0872	0.5761	0.3106
ss	1.0971	0.5794	0.1876
sn_ss	1.0134	0.5759	0.3503
sn (single source)	1.7635	-	-
ss (single source)	3.0512	-	-

TABLE 56: Transfer learning ground truth with MOTS prediction for sn_ss=dp on NYUD.

Ground Truth (dp_sn=ss)			Prediction
Metric / Source	mIoU (fe)	mIoU (ft)	MOTS
dp	0.1367	0.3838	0.1801
sn	0.1307	0.3807	0.1740
dp_sn	0.1677	0.3836	0.1994
dp (single source)	0.0068	-	-
sn (single source)	0.0070	-	-

TABLE 57: Transfer learning ground truth with MOTS prediction for dp_sn=ss on NYUD.

Ground Truth (dp_ss=sn)			Prediction
Metric / Source	RMSE (fe)	RMSE (ft)	MOTS
dp	37.7339	28.463	0.3548
ss	43.0672	28.520	0.1869
dp_ss	37.0614	28.414	0.3872
dp (single source)	63.4431	-	-
ss (single source)	61.6498	-	-

TABLE 58: Transfer learning ground truth with MOTS prediction for dp_ss=sn on NYUD.

On **NYUD** with the source-target combinations sn_ss=dp and dp_ss=sn the rankings of the MOTS prediction are equal to the rankings of the ground truth transfers for the fe and ft approaches (Tables 56 and 58). In case of dp_sn=ss, the MOTS prediction ranking is equal to the ranking of the ground truth from fe; however, the first and second ranks differ between MOTS prediction and ft ground truth. In this case however, the mIoU scores of the first and second ranks differ only in the third decimal point (Table 57). The difference in the result of TL when switching between the two sources is therefore very small.
It can be seen that for all three source-target combinations of NYUD, the best performance is given by the case of transferring from multiple sources (Tables 56, 57 and 58). Comparing the cases where only one branch from the MTI-Net is used as the source to the single-source transfers created by using the single-source transfers from the baseline as explained in Section 5.4.3 (last two rows of Tables 56, 57 and 58), the single-source transfers using the baselines are outperformed in every case. This guides us to the assumption that even when using only one task-head from the MTL architecture for TL, the performance of the transfer is better. Our explanation is that during the joint training of multiple tasks in the multi-task architecture, complementary information from different tasks is learned, which improves the TLP. This effect also occurs when just one task-head from the multi-task architecture is used for the source of the transfer. For the NYUD dataset the MOTS prediction therefore yields a nearly perfect prediction.
Regarding the results using **Pascal VOC**, in the case of ft with the target task ss (Table 59), MOTS predicted the first best transfers correctly and therefore answered the question of which combination sources should be used to obtain the best transfer. In the case of fe all the first three best sources for TL were predicted correctly. However, the cases of hp (Table 61) and sn as target tasks (Table 60) were not predicted

Ground Truth (hp_sn_sl=ss)			Prediction
Source \ Metric	mIoU (fe)	mIoU (ft)	MOTS
hp	0.2191	0.6333	0.4644
sn	0.1992	0.6295	0.4246
sl	0.1135	0.6270	0.3562
hp_sn	0.2702	0.6321	0.5135
hp_sl	0.2383	0.6338	0.5039
sn_sl	0.2294	0.6302	0.4861
hp_sn_sl	0.2790	0.6361	0.5397
hp (single source)	0.0354	-	-
sn (single source)	0.0320	-	-
sl (single source)	0.0268	-	-

TABLE 59: Transfer learning ground truth with MOTS prediction for hp_sn_sl=ss on Pascal VOC.

Ground Truth (hp_sl_ss=sn)			Prediction
Source \ Metric	RMSE (fe)	RMSE (ft)	MOTS
hp	28.5435	18.5929	-0.0028
sl	28.2802	18.5907	0.0028
ss	28.7688	18.5120	-0.0015
hp_sl	27.4432	18.6192	0.0037
hp_ss	27.6783	18.6262	0.0057
sl_ss	27.6676	18.5907	0.0025
hp_sl_ss	26.9392	18.6030	0.0037
hp (single source)	98.3698	-	-
ss (single source)	50.2263	-	-
sl (single source)	91.5639	-	-

TABLE 60: Transfer learning ground truth with MOTS prediction for hp_sl_ss=sn on Pascal VOC.

Ground Truth (sn_sl_ss=hp)			Prediction
Source \ Metric	mIoU (fe)	mIoU (ft)	MOTS
sn	0.3401	0.5984	-0.0021
sl	0.3139	0.6020	-0.0042
ss	0.3965	0.6021	-0.0048
sn_sl	0.3890	0.6004	-0.0019
sn_ss	0.4299	0.6008	-0.0029
sl_ss	0.4163	0.6013	-0.0050
sn_sl_ss	0.4340	0.6002	-0.0027
ss (single source)	0.0847	-	-
sn (single source)	0.1133	-	-
sl (single source)	0.1142	-	-

TABLE 61: Transfer learning ground truth with MOTS prediction for sn_sl_ss=hp on Pascal VOC.

Ground Truth (hp_sn_ss=sl)			Prediction
Metric / Source	mIoU (fe)	mIoU (ft)	MOTS
hp	0.0566	0.0362	0.4287
sn	0.1035	0.0341	0.3431
ss	0.0517	0.0301	0.4559
hp_sn	0.0721	0.0300	0.4594
hp_ss	0.0434	0.0336	0.4856
sn_ss	0.0660	0.0299	0.4856
hp_sn_ss	0.0526	0.0362	0.4994
hp (single source)	0.1869	-	-
ss (single source)	0.1871	-	-
sn (single source)	0.1869	-	-

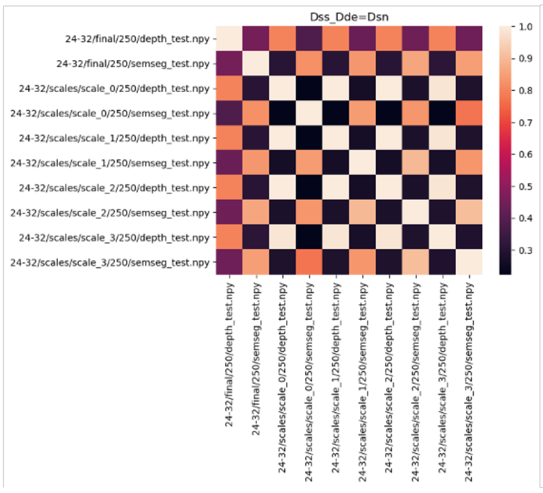
TABLE 62: Transfer learning ground truth with MOTS prediction for hp_sn_ss=sl on Pascal VOC.

correctly by MOTS. The reason is that the MOTS's predictions of all different source combinations are nearly zero, suggesting a poor prediction of the target RDM from the source RDM. In such cases, the MOTS-based rankings are not reliable.
Looking at the results for the target task sl it becomes clear that the mIoU values for all transfers using fe as well as ft are implausibly low, with a maximum of 10%, while the minimum of the transfers using the single-sources as a baseline achieves 18.69%. Due to time constraints, the reason for this behavior could not be further evaluated in this thesis. However, it is reasonable to assume that a bug occurred during the training on target task sl. Therefore, no statement can be made about the functionality and quality of MOTS on the basis of the results using target task sl. For this reason, the results of MOTS for target task saliency are not taken into account in the further analysis. In comparing the results of the transfers from different source combinations using fe it is clear that all the source task combinations from the MTI-Net perform much better than the single-source tasks from the baseline single-source model (last three rows of Table 61), while the best transfer is provided by the three-source task combination sn, sl and ss. A possible reason that the single-source task combinations from the MTI-Net (first three rows of Table 61: sn, sl and ss) are better than the single-source baselines is that in multi-task training, the task interactions improve the overall TLP for the target task.

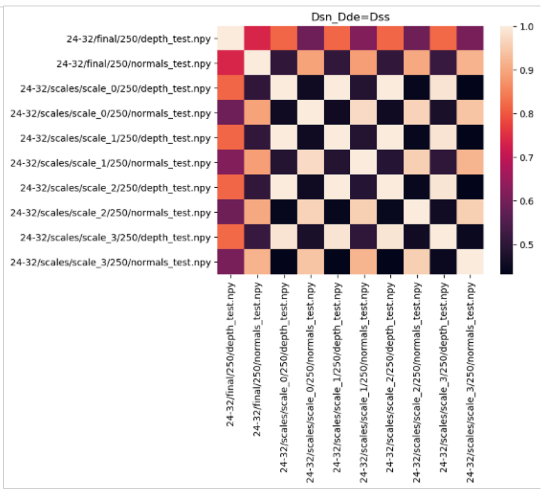
5.4.12 Correlation between MTI-Net Layers (Yannic)

The results described in Section 5.4.11 showed among others that using just one single-task decoder head from the MTI-Net for single-source transfers led to better TL results than using a single-task architecture for that transfer. It must be mentioned as was already described in Section 5.4.8 that we used the last layer of the MTI-Net architecture to do transfers (see Table 31) since the best results were obtained here. We asked ourselves the question if there was another reason for the better performance of MOTS on the final layers over the scaling layers of the MTI-Net architecture.

To find an answer to that question, a correlation matrix between the different feature maps (tasks) at certain layers, that were available for the selection of feature extraction was plotted. The location of the MTI-Net layers that were used for feature extraction is showcased in Table 31 with the following definition of the layers:

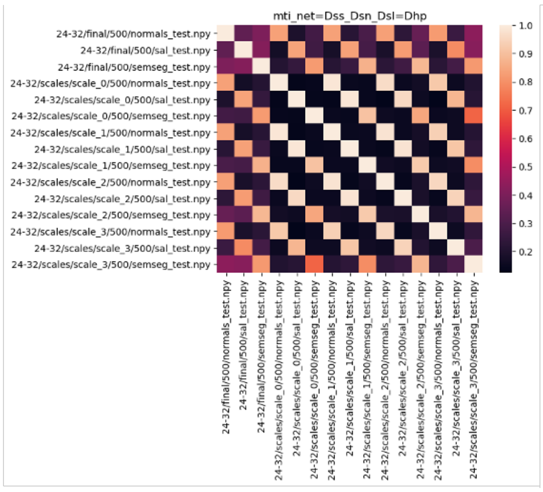


(a) On tasks semantic segmentation and depth estimation

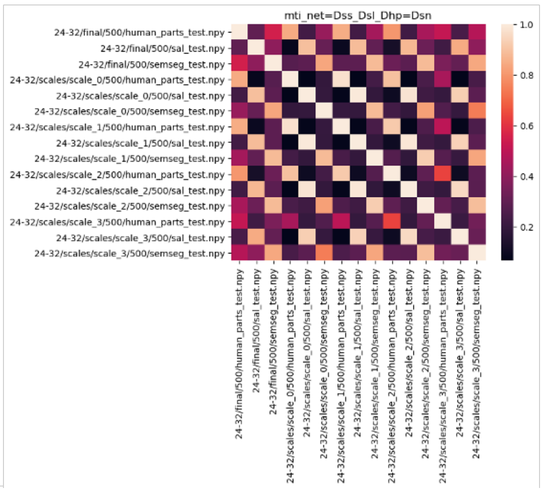


(b) On tasks surface normals and depth estimation

FIGURE 33: Layer correlation on NYUD with layers final, scale 1, scale 2, scale 3 and scale 4 of the MTI-Net with the final layers being at the bottom left and top right corner



(a) On tasks semantic segmentation, surface normals and saliency



(b) On tasks semantic segmentation, saliency and human parts

FIGURE 34: Layer correlation on Pascal VOC with layers final, scale 1, scale 2, scale 3 and scale 4 of the MTI-Net with the final layers being at the bottom left and top right corner

- final layer of the MTI-Net architecture
- scale 3, which is the MTI-Net scale layer, processing images with a resolution size of 1/4
- scale 2, which is the MTI-Net scale layer, processing images with a resolution size of 1/8

- scale 1, which is the MTI-Net scale layer, processing images with a resolution size of 1/16
- scale 0, which is the MTI-Net scale layer, processing images with a resolution size of 1/32

A certain pattern was identified and is partly visible in Figure 33 on NYUD for two source-task combinations and in Figure 34 on Pascal VOC for three source-task combinations. All plots show a higher correlation between different and identical tasks on different layers when at least one final layer was involved (the areas in the bottom left and top right corners of the correlation matrix are lighter and belong to the final layer). That is because all tasks were concatenated in the final layer and thus include task-based information from multiple tasks to a higher degree while this is not the case for the tasks at the scale layers. It appears that a higher TLP with only one task on the MTI-Net using the final layer for feature extraction, promises better results than using a single-task baseline model for transfer due to the fact of implicit shared information from several tasks. This observation strengthens the result, that was presented in 5.4.11 with regards to the higher TLP on single-task decoder heads from a MTL architecture compared to that of a ST model. It further enforces the prediction performance of MOTS, which additionally may be used for the selection of the best layer in a MTL architecture due to the fact, that it accurately suggested the final layers to be used to do TL-ranking predictions on MTI-Net.

Chapter 6

Discussion and Conclusion (Daniel)

In this chapter, we first discuss MOTS regarding the three analyzed aspects, MOTS with single-sources, MOTS with multiple sources and MOTS with multiple tasks. Afterwards, we discuss the limitations of this work and propose some ideas for future research projects.
Finally, we come to a conclusion about our approach and answer our guiding research questions.

6.1 MOTS with Single Sources

In the single-source transfer experiments, MOTS showed a similar high correlation (+0.002 more) to the Taskonomy ground truth as the previous state-of-the art method DDS (Table 63) when 200 images were used to generate the RDMs, as in the DDS paper [40]. In the comparison of DDS and MOTS using 1,800 images and the distance function cosine, MOTS outperformed DDS by 0.009 in terms of higher correlation with the Taskonomy ground truth. In the single-source transfer experiment on Pascal VOC (Table 64, middle column) where the RDMs were generated using Pascal VOC images and the target task was Pascal VOC, MOTS showed a slightly lower correlation with ground truth than DDS, but both provided a high correlation of about 0.8 with Taskonomy ground truth. In the domain transfer experiment (Table 64, rightmost column) where the Taskonomy images were used to generate the RDMs but the target task was Pascal VOC, MOTS showed a correlation of 0.593, which is significantly lower than in the first Pascal VOC experiment. Nevertheless, MOTS outperformed DDS [40], which in this case was 0.539 (f=cosine) and 0.581 (f=Laplacian). MOTS thus dominates the previous state-of-the-art method DDS [40] in two out of three single-source experiments. Although the results are only slightly better, MOTS shows that it already works better in the single-source case than previous methods.

6.2 MOTS with Multiple Sources

The particular strength of MOTS compared to DDS is that MOTS can predict the TLP of multiple sources. To the best of our knowledge, there are no alternative prediction methods for Taskonomy TLP prediction of multiple sources against which MOTS could be benchmarked. Therefore, we compared the correlation of MOTS of single- and double-source transfers with the Taskonomy ground truth with the correlation of DDS of single-source transfers with the Taskonomy ground truth. Compared to DDS's result of 0.871 correlation, MOTS achieved 0.904 correlation with single- and h]

Method	Correlation
Taskonomy Winrate [83]	0.988
Taskonomy affinity [83]	1
saliency [59]	0.605
DeepLIFT [59]	0.681
ϵ-LRP [59]	0.682
RSA [17]	0.777
DDS [40]	0.86
MOTS (200)	0.862
DDS(f=cosine) [40] (train set)	0.862
DDS(f=Laplacian) [40] (train set)	0.863
DDS(f=cosine) [40] (test set)	0.871
DDS(f=Laplacian) [40] (test set)	0.868
MOTS (f=cosine)	**0.880**

TABLE 63: **Correlation of single-source transfer learning on Taskonomy dataset** with 16 Taskonomy tasks. We compare MOTS to state-of-the-art methods using 200 images (white rows). Grey rows: we compare MOTS using 1,800 images for train and 1,800 for test to DDS using cosine as well as Laplacian similarity with the 1,800 train set and the 1800 test set.

Method	Pascal VOC	Taskonomy
DDS (f=cosine)	0.789	0.539
DDS (f=Laplacian)	**0.801**	0.581
MOTS	0.799	**0.593**

TABLE 64: **Correlation of single-source transfer learning for Pascal VOC semantic segmentation**. For both, MOTS and DDS, the RDMs were computed with 1,800 images. We created RDMs using images from different datasets (Pascal VOC and Taskonomy).

double-sources, beating DDS by almost 3% percent (see Table 65). It is surprising that MOTS achieved only a low correlation of 0.4390 with ground truth in the case where only two source transfers were used. However, the poor performance can be attributed to certain target tasks in the Taskonomy dataset, all of which have a very low variance of TLP values with this target task. This means that the choice of the source task for transfers to these target tasks with bad predictions has little significance because a similar TLP is achieved with all possible choices for the source task.

Method	Correlation
DDS (f=Laplacian) [40] (train,single)	0.863
DDS (f=cosine) [40] (test,single)	0.871
MOTS (single)	0.880
MOTS (multi)	0.430
MOTS (all)	**0.904**

TABLE 65: **Correlation of multi-source transfer learning on Taskonomy dataset** with 16 Taskonomy tasks. We compare MOTS when using only multiple sources (multi= two source tasks) and all sources (one and two source tasks combined), as well as single-source transfer learning (single) to DDS with single-source transfer learning, in all cases using 1,800 images for computing the RDMs, both for training and testing.

6.3 MOTS with Multiple Tasks

With the application of MOTS to the task-heads of the MTI-Net we showed a specific use case for MOTS. We created ground truth TLP values for two datasets (NYUD and Pascal VOC) and evaluated the MOTS prediction with the created ground truth. The results of this evaluation were two sided: On the one hand, MOTS achieved an almost perfect prediction for all target tasks in the NYUD case, only once confusing the first and second rank. In addition, the ground truth value of the first and second ranks in that case only differ in the third decimal place, which means that the difference between a transfer of the first or second rank is almost the same so MOTS still delivers a very good prediction. On the other hand, MOTS was not convincing in the Pascal VOC experiment. It should be noted that the prediction of a correct ranking is significantly more difficult in the Pascal VOC case than in the NYUD case since only three source combinations are evaluated in the NYUD transfers, whereas in Pascal VOC there are seven source combinations per target task. Nevertheless, it can be seen that in the cases of the target tasks sn and hp having near zero values, and partly slightly negative MOTS values were generated. This means that the target RDMs are poorly predicted by the source RDM using MOTS linear regression in these cases. Although some experiments seem promising, MOTS lacked a general reliability in our multi-task experiment. Therefore, further experiments are necessary to prove or disprove MOTS applicability for selecting the optimal source for task-heads from a MTL architecture.
An interesting observation from the experiments on NYUD and Pascal VOC is that in all cases (target task sl is not considered because implausible results) of fe as a TL method, the source combination with the most possible task heads (NYUD: two task heads, Pascal VOC: three task heads) achieves the best TLP and outperforms all other transfers based on the MTI-Net and the single-task baseline transfers.

6.4 Limitations

There are some limitations regarding the interpretability of the results of this thesis. The experiments evaluating MOTS were limited because there is no large-scale ground truth for multiple source TL against which to evaluate MOTS.
The use case of predicting the TLP of the combinations of task-decoder heads in a MTL architecture was only evaluated with the MTI-Net. It remains to be evaluated whether the MOTS prediction behaves differently with a different multi-task architecture.

6.5 Ideas for Future Work

A few ideas for further research arose from the project:

- For better evaluability of multi-source transfer prediction methods, a large-scale ground truth dataset is needed, especially one that includes the TL using different combinations of task-heads as sources. A project providing a ground truth dataset for the TLP from transfers from different taks-heads of state-of-the-art MTL architectures would also be highly beneficial.
- The configuration of the MTI-Net was mainly kept fixed and close to the provided configuration of the MTI-Net framework. This was done on purpose to allow an easy comparability. However, it might be interesting for future studies to change training configurations such as the optimizer and use other visual tasks and another backbone. An obvious step would be to repeat the experiments with a Residual neural network (ResNet) instead of the HRNet as the backbone, since ResNet backbone is provided by the MTI-Net implementation.
- As already explained in the limitations section (Section 6.4), the evaluation of MOTS in a multi-task setting was only performed using MTI-Net as the multi-task model. Other multi-task architectures could result in further insights into transferability from different multi-task heads. Interesting candidates for MTL architectures include PAP-Net or PAD-Net.

6.6 Conclusion

In this thesis, we analyzed and evaluated a new method for predicting TLP: MOTS. The main advantage of our method over existing methods is its ability to consider multiple source tasks for the prediction of TLP. To evaluate this methodology, we set several research questions in the introduction. Our main question was: **Does our similarity method promise state-of-the-art results in TL when using a linear fit between source and target task for single as well as multiple source tasks?**
To answer this main question, we divided it in three further subquestions, which will be answered before coming back to our main research question.
The first subquestion is: **How well does MOTS perform in single-source TLP prediction?** As already explained in the discussion section, MOTS shows not only a similar performance to state-of-the-art methods; it surpasses them in two-thirds of the results.
Our second guiding question tackles MOTS' main feature, its application to multiple sources for TL prediction: **How well does MOTS perform in multi-source TLP**

prediction?
Although MOTS cannot be benchmarked in multi-source TL against other prediction methods, it can be evaluated against the brute-force approach of Taskonomy. The correlation of MOTS predictions with Taskonomy ground truth using single-source and multiple-source transfers improved by 0.024 from MOTS using just single-source transfers to MOTS using single- and multiple-source transfers.
The third guiding question is: **How well does MOTS perform in TLP prediction using the task-decoder heads of a mulit-task architecture as sources?** This question has to be answered from different perspectives. MOTS results for the prediction of the optimal source combination of task-heads were nearly optimal; however, MOTS showed only limited predictive power in a similar experiment with the Pascal VOC dataset.
Does our similarity method promise state-of-the-art results in TLP prediction when using a linear fit between source and target task for single- as well as multi-source tasks?
In summary, our main question can be answered as follows: In terms of predictive power in the single-source and multi-source cases, MOTS is convincing and surpasses the state-of-the-art method DDS in the majority of cases. Therefore, predicting the target RDMs using the source RDMs via a linear regression is a promising approach. Whether this approach is suited for the prediction of the optimal source combination of task-decoder heads in a multi-task architecture could not be finally proven and needs further analysis.

Bibliography

[1] Héctor Martínez Alonso and Barbara Plank. *When is multitask learning effective? Semantic sequence prediction under varying data conditions*. URL: https://arxiv.org/pdf/1612.02251.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. URL: https://arxiv.org/pdf/2003.05991.

[3] Herbert Bay, Tinne Tuytelaars, and Luc van Gool. "SURF: Speeded Up Robust Features". In: *Computer vision - ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Berlin: Springer, 2006, pp. 404–417. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_32.

[4] *Beam search*. 10/26/2004. URL: https://personalpages.bradley.edu/~chris/searches.html.

[5] Joachim Bingel and Anders Søgaard. *Identifying beneficial task relations for multi-task learning in deep neural networks*. URL: https://arxiv.org/pdf/1702.08303.

[6] John Blitzer, Mark Dredze, and Fernando Pereira. "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 440–447. URL: https://aclanthology.org/P07-1056.

[7] Marco Buzzelli, Simone Bianco, and Gianluigi Ciocca. "Combining Saliency Estimation Methods". In: *Image analysis and processing - ICIAP 2019*. Ed. by Elisa Ricci et al. Vol. 11752. LNCS sublibrary. SL 6, Image processing, computer vision, pattern recognition, and graphics. Cham, Switzerland: Springer, 2019, pp. 326–336. ISBN: 978-3-030-30644-1. DOI: 10.1007/978-3-030-30645-8_30.

[8] B. Chen. "Why Rectified Linear Unit (ReLU) in Deep Learning and the best practice to use it with TensorFlow". In: *Towards Data Science* (1/11/2021). URL: https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow-e9880933b7ef.

[9] "Coefficient of Determination". In: *The Concise Encyclopedia of Statistics*. Springer, New York, NY, 2008, pp. 88–91. DOI: 10.1007/978-0-387-32833-1_62. URL: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-32833-1_62.

[10] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. URL: http://arxiv.org/pdf/2009.09796v1.

[11] Wenyuan Dai et al. "Co-clustering based classification for out-of-domain documents". In: *KDD 2007*. Ed. by Pavel Berkhin. New York, New York, USA: ACM Press, 2007, p. 210. ISBN: 9781595936097. DOI: 10.1145/1281192.1281218.

[12] Wenyuan Dai et al. "Self-taught clustering". In: *Proceedings, Twenty-fifth International Conference on Machine Learning*. Ed. by Andrew K. McCallum and Sam Roweis. Helsinki, Finland: University of Helsinki], 2008, pp. 200–207. ISBN: 9781605582054. DOI: 10.1145/1390156.1390182. URL: https://www.cse.ust.hk/~qyang/Docs/2008/dwyakicml.pdf.

[13] Kenneth Dawson-Howe. *A practical introduction to computer vision with OpenCV*. Online-Ausg. Chichester, England: Wiley, 2014. ISBN: 9781118848784. URL: http://site.ebrary.com/lib/alltitles/Doc?id=10856780.

[14] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 6/20/2009 - 6/25/2009, pp. 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[15] Adit Deshpande. *A Beginner's Guide To Understanding Convolutional Neural Networks*. 2.04.2021. URL: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

[16] Carl Doersch and Andrew Zisserman. *Multi-task Self-Supervised Visual Learning*. URL: https://arxiv.org/pdf/1708.07860.

[17] Kshitij Dwivedi and Gemma Roig. "Representation Similarity Analysis for Efficient Task taxonomy & Transfer Learning". In: *CoRR* abs/1904.11740 (2019). arXiv: 1904.11740. URL: http://arxiv.org/abs/1904.11740.

[18] Y. Escoufier. "The Duality Diagram: A Means for Better Practical Applications". In: *Develoments in Numerical Ecology*. Ed. by Pierre Legendre and Louis Legendre. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 139–156. ISBN: 978-3-642-70882-4. DOI: 10.1007/978-3-642-70880-0_3.

[19] Mark Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0733-5.

[20] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4.

[21] H. Faris, I. Aljarah, and S. Mirjalili. *Evolutionary Data Clustering: Algorithms and Applications*. Algorithms for intelligent systems. Springer, 2021. ISBN: 9789813341913. URL: https://books.google.de/books?id=20wfEAAAQBAJ.

[22] Cornelia Fermüller and Yiannis Aloimonos. "Vision and action". In: *Image and Vision Computing* 13.10 (1995), pp. 725–744. ISSN: 02628856. DOI: 10.1016/0262-8856(95)98754-H.

[23] Ernest H. Forman and Saul I. Gass. "The Analytic Hierarchy Process—An Exposition". In: *Operations Research* 49.4 (2001), pp. 469–486. ISSN: 0030-364X. DOI: 10.1287/opre.49.4.469.11231.

[24] GitHub. *ludvb/batchrenorm: Batch Renormalization in Pytorch*. 14.09.2021. URL: https://github.com/ludvb/batchrenorm.

[25] GitHub. *taskonomy/results at master · StanfordVL/taskonomy*. 14.09.2021. URL: https://github.com/StanfordVL/taskonomy.

[26] Clement Godard et al. "Digging Into Self-Supervised Monocular Depth Estimation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 3827–3837. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00393. URL: https://openaccess.thecvf.com/content_ICCV_2019/papers/Godard_Digging_Into_Self-Supervised_Monocular_Depth_Estimation_ICCV_2019_paper.pdf.

[27] Alexander Goldenshluger and Assaf Zeevi. "The Hough transform estimator". In: *The Annals of Statistics* 32.5 (2004). ISSN: 0090-5364. DOI: 10.1214/009053604000000760.

[28] Curt Hagquist and Magnus Stenbeck. "Goodness of Fit in Regression Analysis – R2 and G2 Reconsidered". In: *Quality and Quantity* 32.3 (Aug. 1998), pp. 229–245.

[29] Kaiming He et al. *Deep Residual Learning for Image Recognition*. URL: https://arxiv.org/pdf/1512.03385.

[30] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[31] T. Huang. *Computer Vision: Evolution And Promise*. 1996. DOI: 10.5170/CERN-1996-008.21.

[32] Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. "What makes ImageNet good for transfer learning?" In: *CoRR* abs/1608.08614 (2016). arXiv: 1608.08614. URL: http://arxiv.org/abs/1608.08614.

[33] Andrew Hynes and Stephen Czarnuch. "Human Part Segmentation in Depth Images with Annotated Part Positions". In: *Sensors (Basel, Switzerland)* 18.6 (2018). DOI: 10.3390/s18061900.

[34] "Integer Programming: Branch and Bound Methods". In: *SpringerReference*. Berlin/Heidelberg: Springer-Verlag, 2011. URL: http://web.tecnico.ulisboa.pt/mcasquilho/compute/_linpro/TaylorB_module_c.pdf.

[35] Ebrahim Karami, Mohamed Shehata, and Andrew Smith. *Image Identification Using SIFT Algorithm: Performance Analysis against Different Image Deformations*. URL: https://arxiv.org/pdf/1710.02728.

[36] "Pearson's Correlation Coefficient". In: *Encyclopedia of Public Health*. Ed. by Wilhelm Kirch. Dordrecht: Springer Netherlands, 2008, pp. 1090–1091. ISBN: 978-1-4020-5614-7. DOI: 10.1007/978-1-4020-5614-7_2569. URL: https://doi.org/10.1007/978-1-4020-5614-7_2569.

[37] S. Kornblith, J. Shlens, and Q. V. Le. "Do Better ImageNet Models Transfer Better?" In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2019, pp. 2656–2666. DOI: 10.1109/CVPR.2019.00277. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00277.

[38] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. "Representational similarity analysis - connecting the branches of systems neuroscience". In: *Frontiers in systems neuroscience* 2 (2008), p. 4. DOI: 10.3389/neuro.06.004.2008.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* 25 (8436).

[40] Kshitij Dwivedi et al. "Duality Diagram Similarity: a generic framework for initialization selection in task transfer learning". In: (2020).

[41] Lawrence G. Roberts. *Machine Perception of Three-Dimensional Solids*. 1963. ISBN: 0-8240-4427-4. URL: https://www.researchgate.net/publication/220695992_Machine_Perception_of_Three-Dimensional_Solids.

[42] Erik G. Learned-Miller. "Introduction to Computer Vision". In: (2011). URL: https://people.cs.umass.edu/~elm/Teaching/Docs/IntroCV_1_19_11.pdf.

[43] Yann LeCun et al. "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision*. Ed. by David A. Forsyth et al. Vol. 1681. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. ISBN: 978-3-540-66722-3. DOI: 10.1007/3-540-46805-6_19.

[44] Fei-Fei Li, Justin Johnson, and Serena Yeung. *cs231: Detection and Segmentation: Lecture 11*. URL: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf.

[45] Niall O' Mahony et al. "Deep Learning vs. Traditional Computer Vision". In: 943 (2020). DOI: 10.1007/978-3-030-17795-9. URL: http://arxiv.org/pdf/1910.13796v1.

[46] Ishan Misra et al. *Cross-stitch Networks for Multi-task Learning*. URL: https://arxiv.org/pdf/1604.03539.

[47] Guido Montufar. *Restricted Boltzmann Machines: Introduction and Review*. URL: https://arxiv.org/pdf/1806.07066.

[48] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[49] *PASCAL VOC2012 Database Statistics*. http://host.robots.ox.ac.uk/pascal/VOC/voc2012/dbstats.html. Accessed: 2021-12-17.

[50] Christian P. Robert, Nicolas Chopin, and Judith Rousseau. "Harold Jeffreys's Theory of Probability Revisited". In: *Statistical Science* 24.2 (2009). ISSN: 0883-4237. DOI: 10.1214/09-STS284.

[51] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection". In: *Computer vision - ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Berlin: Springer, 2006, pp. 430–443. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_34.

[52] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. URL: http://arxiv.org/pdf/1706.05098v1.

[53] German Sharabok. *Why Deep Learning Uses GPUs?. And why you should too... | by German Sharabok | Towards Data Science*. 2020. URL: https://towardsdatascience.com/why-deep-learning-uses-gpus-c61b399e93a0.

[54] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network". In: *Physica D: Nonlinear Phenomena* 404.8 (2020), p. 132306. ISSN: 01672789. DOI: 10.1016/j.physd.2019.132306. URL: https://arxiv.org/pdf/1808.03314.

[55] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning Important Features Through Propagating Activation Differences”. In: *PMLR 70:3145-3153* (). URL: https://arxiv.org/pdf/1605.01713.pdf.

[56] Nathan Silberman and Rob Fergus. “Indoor scene segmentation using a structured light sensor”. In: *2011 IEEE International Conference on Computer Vision workshops (ICCV workshops 2011)*. Piscataway, NJ: IEEE, 2011, pp. 601–608. ISBN: 978-1-4673-0063-6. DOI: 10.1109/ICCVW.2011.6130298.

[57] Nathan Silberman et al. “Indoor Segmentation and Support Inference from RGBD Images”. In: *Computer Vision – ECCV 2012*. Ed. by David Hutchison et al. Vol. 7576. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760. ISBN: 978-3-642-33714-7. DOI: 10.1007/978-3-642-33715-4_54.

[58] Bhupesh Kumar Singh. “Evaluation of Genetic Algorithm as Learning System in Rigid Space Interpretation”. In: *Handbook of research on novel soft computing intelligent algorithms*. Ed. by Pandian Vasant. Advances in computational intelligence and robotics (ACIR) book series, 2327-0411. Hershey, PA: Information Science Reference, 2014, pp. 475–510. ISBN: 9781466644502. DOI: 10.4018/978-1-4666-4450-2.ch016.

[59] Jie Song et al. *Deep Model Transferability from Attribution Maps*. URL: http://arxiv.org/pdf/1909.11902v2.

[60] Jie Song et al. “DEPARA: Deep Attribution Graph for Deep Knowledge Transferability”. In: *Proceedings, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Los Alamitos, California: IEEE Computer Society, Conference Publishing Services, 2020, pp. 3921–3929. ISBN: 978-1-7281-7168-5. DOI: 10.1109/CVPR42600.2020.00398. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Song_DEPARA_Deep_Attribution_Graph_for_Deep_Knowledge_Transferability_CVPR_2020_paper.pdf.

[61] “Spearman Rank Correlation Coefficient”. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 502–505. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_379. URL: https://doi.org/10.1007/978-0-387-32833-1_379.

[62] Trevor Standley et al. *Which Tasks Should Be Learned Together in Multi-task Learning?* URL: https://arxiv.org/pdf/1905.07553.

[63] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. “Learning Task Relatedness in Multi-Task Learning for Images in Context”. In: *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. Ed. by Abdulmotaleb El Saddik. ACM Digital Library. New York,NY,United States: Association for Computing Machinery, 2019, pp. 78–86. ISBN: 9781450367653. DOI: 10.1145/3323873.3325009.

[64] Richard Szeliski. *Computer vision: Algorithms and applications*. Texts in computer science. London: Springer, 2011. ISBN: 9781848829350. DOI: 10.1007/978-1-84882-935-0. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10421311.

[65] Chuanqi Tan et al. “A Survey on Deep Transfer Learning”. In: *CoRR* abs/1808.01974 (2018). arXiv: 1808.01974. URL: http://arxiv.org/abs/1808.01974.

[66] *Taskonomy: Disentangling Task Transfer Learning*. https://github.com/StanfordVL/taskonomy. Accessed: 2021-12-17.

[67] Keras Team. *Keras documentation: Image segmentation metrics*. 12/3/2021. URL: `https://keras.io/api/metrics/segmentation_metrics/`.

[68] Keras Team. *Keras documentation: Monocular depth estimation*. 12/3/2021. URL: `https://keras.io/examples/vision/depth_estimation/`.

[69] TensorFlow. *Transfer learning and fine-tuning*. 12/17/2021. URL: `https://www.tensorflow.org/tutorials/images/transfer_learning`.

[70] *The Beginner's Guide to Semantic Segmentation*. 12/18/2021.

[71] Ekin Tiu. *Metrics to Evaluate your Semantic Segmentation Model*. 12/18/2021. URL: `https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2`.

[72] *torch.clamp — PyTorch 1.9.0 documentation*. 14.09.2021. URL: `https://pytorch.org/docs/stable/generated/torch.clamp.html`.

[73] Simon Vandenhende, Stamatios Georgoulis, and Luc van Gool. *MTI-Net: Multi-Scale Task Interaction Networks for Multi-Task Learning*. URL: `http://arxiv.org/pdf/2001.06902v5`.

[74] Simon Vandenhende et al. *Branched Multi-Task Networks: Deciding What Layers To Share*. URL: `https://arxiv.org/pdf/1904.02920`.

[75] Simon Vandenhende et al. *Multi-Task Learning for Dense Prediction Tasks: A Survey*. URL: `http://arxiv.org/pdf/2004.13379v2`.

[76] Jiguang Wang. "Pearson Correlation Coefficient". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, 2013, pp. 1671–1671. ISBN: 978-1-4419-9863-7. DOI: `10.1007/978-1-4419-9863-7_372`. URL: `https://doi.org/10.1007/978-1-4419-9863-7_372`.

[77] Jingdong Wang et al. *Deep High-Resolution Representation Learning for Visual Recognition*. URL: `https://arxiv.org/pdf/1908.07919`.

[78] Jinjiang Wang et al. "Deep learning for smart manufacturing: Methods and applications". In: *Journal of Manufacturing Systems* 48 (2018), pp. 144–156. ISSN: 02786125.

[79] Xiaolong Wang, David F. Fouhey, and Abhinav Gupta. *Designing Deep Networks for Surface Normal Estimation*. URL: `https://arxiv.org/pdf/1411.4958`.

[80] Victor Wiley and Thomas Lucas. "Computer Vision and Image Processing: A Paper Review". In: *International Journal of Artificial Intelligence Research* 2.1 (2018), p. 22. DOI: `10.29099/ijair.v2i1.42`.

[81] Dan Xu et al. *PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing*. URL: `http://arxiv.org/pdf/1805.04409v1`.

[82] Qiang Yang et al. "Heterogeneous Transfer Learning for Image Clustering via the SocialWeb". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 1–9. URL: `https://www.aclweb.org/anthology/P09-1001`.

[83] Amir Zamir et al. *Taskonomy: Disentangling Task Transfer Learning*. URL: `https://arxiv.org/pdf/1804.08328`.

[84] Amir R. Zamir et al. "Supplementary Material, Taskonomy: Disentangling Task Transfer Learning". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 18.06.2018 - 23.06.2018, pp. 3712–3722. ISBN: 978-1-5386-6420-9. DOI: `10.1109/CVPR.2018.00391`.

[85] Zhenyu Zhang et al. *Pattern-Affinitive Propagation across Depth, Surface Normal and Semantic Segmentation*. URL: `https://arxiv.org/pdf/1906.03525`.