Universal Representations: Towards Multi-Task Learning & Beyond

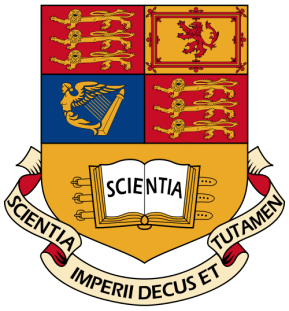
Thesis by

Liu, Shikun

In Partial Fulfillment of the Requirements

for the Degree of

Master of Research in Advanced Computing



Imperial College of Science, Technology and Medicine

London, United Kingdom

(Completed September 7, 2018)

To all the kids who loved day-dreaming and fantasising.

Butterflies, 1950 - M.C. Escher

Acknowledgments

I would like to acknowledge a number of people that supported, in various ways, to the creation of this work.

First and foremost, I would like to express my greatest gratitude towards my supervisor *Andrew Davison* for accepting me as his research student. Andy gave me a wonderful opportunity to work in the *Dyson Robotics Lab* with much academic freedom, and he helped me on various academic affairs as much as he can. I could not have hoped for an advisor with a greater commitment to research excellence and with a deep amicability and humility. I was especially grateful for his advise on doing research which had a direct influence on my scientific reserach capacity:

"*The right thing is to concentrate on the work that you think is intrinsically good and interesting to you, and keep working on that for a long time, even if the rewards don't come immediately. Eventually if what you are doing is really good then people will pay attention.* "

I would also like to thank my co-advisor *Edward Johns* who introduced the interesting territory of robotics manipulation and reinforcement learning. I would like to appreciate his sufficient patience and efforts on the help of my academic writing and his constant encouragements when the research did not go well. Our weekly discussions and brainstorming together with Andy were of key importance to formulate most of the projects covered in this work.

Furthermore, I would love to thank *Krysia Broda* for all the help during the master of research program I attended at Imperial College. Her timely and useful advise on doctoral studies and frequent chatting without any strings were one of the warmest memories in the clammy weather across the city of London.

This work could not be completed without the additional help from many members in the lab. I would like to credit (in alphabetical order) *Michael Bloesch, Ronald Clark, John McCormac, Stephen James, Zoe Landgraf* and *Shuaifeng Zhi* for teaching me use the computing resources in the lab and providing constructive comments on some of the projects in this work.

I would like to further thank my two lovely flatmates *Xiaoqing Huang* and *Danlin Peng* bearing with my terrible daily routines and many Chinese schoolmates from Imperial College organising multiple dinner parties to feel like home: familiar and friendly. Lastly, I would like to recognise my parents for the financial and moral support and of course my fiancée *Shan Huang* to make me believe in love.

Abstract

Humans are able to solve tasks under many different domains. When facing new tasks, we leverage knowledge acquired from previous experiences, then we accumulate and transfer knowledge to a new task for continual learning. Multi-task learning is a learning paradigm which aims to leverage the shared features learned in a unified framework to improve generalisation in contrast to learning each task independently. It is more efficient not only in terms of memory and inference speed, but also in terms of data, since related tasks may share informative feature representations. In this thesis, we will investigate multi-task learning with a spectrum of different perspectives.

We first give an overview and thus touch upon several major problems in multi-task learning. We introduce some modern approaches on the multi-task network design and discuss the effectiveness of relative task weighting. We then show how, a multi-task network design can be tailored in a way to incorporate both *task-agnostic* and *task-specific* features in a self-supervised and efficient manner. As part of our evaluation of robustness on task weighting, we also propose a novel weighting scheme, which adapts the task weighting over time by considering the rate of change of the loss for each task.

As further introspection into the benefits of multi-task learning, we also designed experiments to investigate how *generalisation* scales with different task complexity, such as an increasing number of semantic classes or training with an increasing number of learning tasks. Lastly, we take a step further, to present an auxiliary learning framework which can automatically generate *knowledge* to improve generalisation. The results offer a promising path towards building machine-generated knowledge and providing a new perspective on generalisation in the regime of deep learning.

Nomenclature

Symbols & Notations

D	dataset
X	dataset inputs
Y	dataset outputs
x_n	single data point
y_n	single data label
$x_n(i, j)$	element at row i column j in data point x_n
$\hat{y}_n$	predicted model output on input x_n
ϵ	a random variable
λ	task weighting
τ	temperature
$f_\omega(\cdot)$	a function f with parameters ω
$\mathcal{N}(\cdot)$	Gaussian (normal) distribution
$\mathcal{B}(\cdot)$	Bernoulli distribution
$\|\|\cdot\|\|$	norm-1 distance
$\|\|\cdot\|\|_2$	norm-2 distance (Euclidean norm)
$[\cdot;\cdot]$	concatenation

Acronyms & Abbreviations

KL	Kullback-Leibler
NN	Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
RL	Reinforcement Learning
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
RAE	Relative Absolute Error
e.g.	exempli gratia (for example)
i.e.	id est (that is)
s.t.	such that
w.r.t.	with respect to

Contents

Introduction and Background I

In this chapter, we introduce and explain background material, mostly the building blocks of deep learning (Section 1.2) which include convolutional neural networks (Section 1.2.1) and recurrent neural networks (Section 1.2.2). We elaborate our motivations and related research areas in multi-task learning (Section 1.3) as well as its applications (Section 1.4). Eventually, we present our research contributions and thesis outline (Section 1.5).

1.1 Machine Intelligence and Computer Vision

The initial concept of *machine (or artificial) intelligence* can be traced back as early as more than half a century ago where Alan Turing published a phenomenal paper - 'Computing Machinery and Intelligence' [Tur09] in which he introduced *Turing Test* to the general public, a test of machine's ability to exhibit intelligent behaviour equivalent to that of a human. In the years following, impressive advances have been made in the field of machine intelligence. One of the most notable achievements was ELIZA [Wei66], a chatbot simulated conversation using pattern matching method, and it was regarded as one of the first programs capable of passing the Turing Test.

Before the AI booming era today, there were two 'AI winter' spanning several decades from the 1970s to the late 1980s mainly because of the high expenses and the limitation of speed and power in legacy computing machines. The AI winter thawed in the early 1990s and has been increasingly well funded till today. However, as long as AI has gone mainstream, some people worried about the progress of AI characterised with hype and alarmism.

The goal of any machine intelligence system is to create *knowledge*, whether learning from experience, or illuminating entirely new domains [Sim95, Kel15]. It follows that a fundamental consideration in machine intelligence is the underlying *theory of knowledge creation*. Obviously, a theory of knowledge shouldn't become an obstruction towards the pursuit of machine intelligence. Just like many major scientific achievements: the electrical and communication apparatus preceded the theory of electromagnetism; the telescope preceded the optics theory. Similarly, many researchers are pursuing machine intelligence without a rigorous explanation of how knowledge is created.

The invention of *multilayer perceptrons* [IL66] (now we called deep learning, refer to Section 1.2) was one representative example of a good learning algorithm lacking a fundamental theory. Many successful practitioners in the early 1980s were hard-pressed to illuminate the theory that underpins their solutions. The rising and breakthrough sparkled in the early 2010s thanks to the major speedup offered by graphics processing units (GPUs) to train large models. Currently, deep learning is the most popular method in the field of machine intelligence and has produced results comparable to and sometimes superior to human experts in many applications [MKS+15, HZRS15].

One of the most important areas in machine intelligence is computer vision, i.e., to teach machines *understand* images and videos in an automatic way. Computer vision is a vibrant field and there are myriad interesting research questions that fall under this big umbrella - knowing what objects are present [SZ14, HZRS16a], their locations [HGDG17], semantic meanings [MHLD16] and spatial relations [SRB+17]. A learning system with such a capability can arguably process a certain level of *intelligence*, and sure step towards an understanding of cognition [RD02].

Going beyond pure scientific discoveries, the recent successes in computer vision have influenced the pop culture, cultivated many commercial applications and different aspects of the infiltration and integration. Human motion capture in gaming and feature films [MG01, Gav99], neural style transfer in a smart phone [GEB16], augmented reality in a virtual enviroment [Azu97, VKP10], and visual SLAM and navigation in a robotic system [DRMS07] are only the tip of the iceberg.

The aim of this work is a further advancement in a general aspect of machine intelligence; more precisely, to accelerate and improve *learning* in an autonomous system.

1.2 Deep Learning

Deep learning (also known as deep neural network or multilayer perceptrons) is a learning system in a *connectionist* paradigm. Unlike conventional methods which we hardcode explicit procedures in a linear fashion, neural networks process information collectively throughout a layer-by-layer architecture composed with computational units (in this case, we called neurons) to learn rich and useful features in a self-supervised manner.

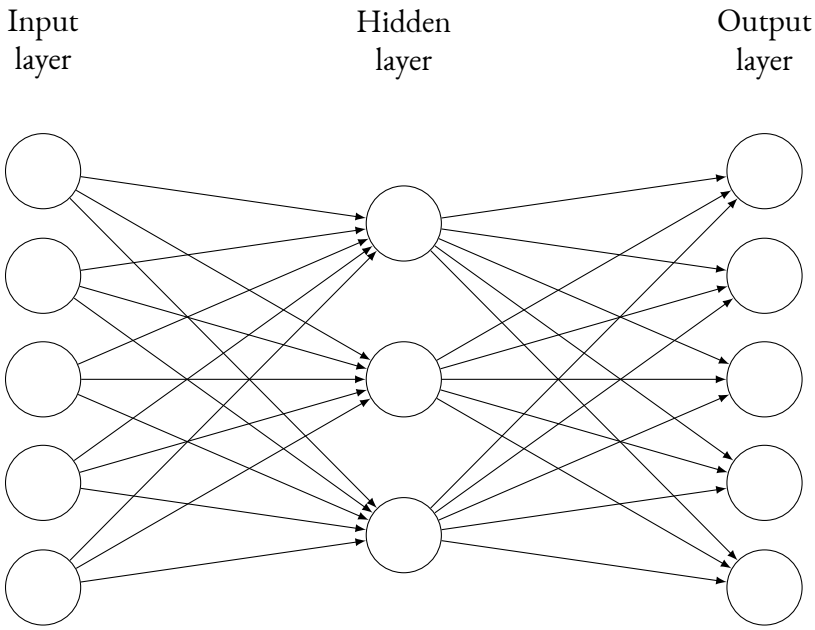


fig. 1.1: A illustration of a simple 3-layer feed-forward neural network architecture.

The ability of learning is achieved by adjusting the *weights* in each connection between two neurons by back-propagating errors using gradient descent. Each neuron taking the input from previous layer is applied to a linear transformation as it flows through a neural network: $f(x) = xW + b$. Each layer is usually further applied to a non-linear activation $\phi(\cdot)$ before sent to the next layer to encourage non-linearities. The most commonly used activation functions are,

$$\text{sigmoid}(z) = \frac{1}{1+e^{-z}}, \quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \text{ReLu}(z) = \max(0, z)^{1}$$

[1]The activation function ReLu (Rectified Linear units) unlike the previous two is not bounded or differentiable. The recent results have argued that ReLu performs a great acceleration to the convergence due to its linearity and unboundedness [KSH12] compared with other activation functions.

In an example of regression, we are given a set of N number of data pairs $D = (x_{1:N}, y_{1:N})$. Our objective is to find network parameters W, b such that minimise the average mean square error over our observed data: $\frac{1}{N}\sum_{i=1}^{N} ||\phi(x_i W_1 + b_1)W_2 + b_2 - y_i||_2^2$ for a 3-layer feed-forward neural network as shown in Figure 1.1.

Each layer in the network hierarchy can be seen as a *building block* of a deep learning architecture. The composition of such blocks embodies the versatility of deep learning models which leads to an infinite number of possibilities on model combinations. Nevertheless, the fundamental understanding of *expressibility* in neural network, i.e. how the architectural priorities of a neural network affect the performance and resulting functions it can compute, is still lacking. The very first results related to this question took a highly theoretical approach to show: a feed-forward neural network can be considered as a *Universal Function Approximator* [HSW89] which is capable of approximating any measurable function to any desired degree of accuracy. Some more recent analytical discoveries from [MPCB14] explained the compositional properties in deep models with piecewise linear activations, where computations on higher layers are effectively replicated by recursive *space folding* operations in all input regions that produced the same output at a given layer. Figure 1.2 offers an illustration of this replication property.

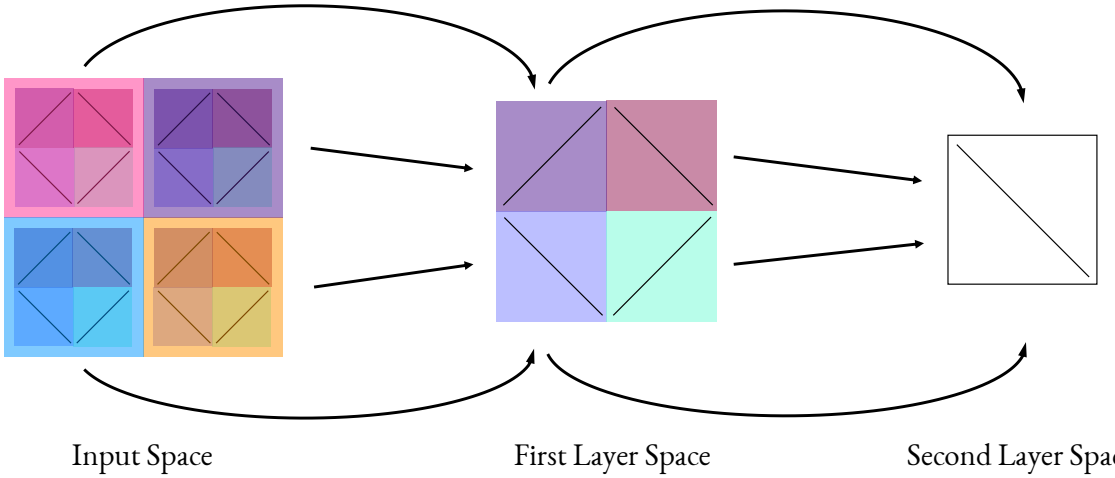


fig. 1.2: A illustration of space folding: how the top-level partitioning (on the right) is replicated to the original input space (on the left). Created by adaptions of original figures from [MPCB14].

Towards a set of more fundamental questions, such as the *measure* of expressibility of deep neural network and complexity bound of network constructions (separation between deep and shallow networks) has largely remained unanswered. A deeper understanding of these issues might begin to draw connections between network expressibility and observed performance [RPK+16], ultimately understand *generalisation* in deep learning.

The performance of different architectures in neural networks varies on different types of data. In the following sections, we will introduce the two most common network structures in deep neural network design: convolutional neural networks which are good at dealing with images (Section 1.2.1) and recurrent neural networks which are good at dealing with series of data (Section 1.2.2). Finally, we refer readers to the textbook [GBCB16] for a more detailed introduction.

1.2.1 Convolutional Neural Network

Convolutional Neural Networks (ConvNets or CNNs) which derive the name from convolution operators are designed mainly for dealing with images. One common convolutional architecture is composed by recursively stacking convolutional layers and pooling layers with optional fully-connected layers are attached at the end of the network depending on the type of task. The first deep convolutional net was introduced by *LeCun, et al* by proposing a network called *LeNet5* [LBBH98] for recognising hand-written postcodes.

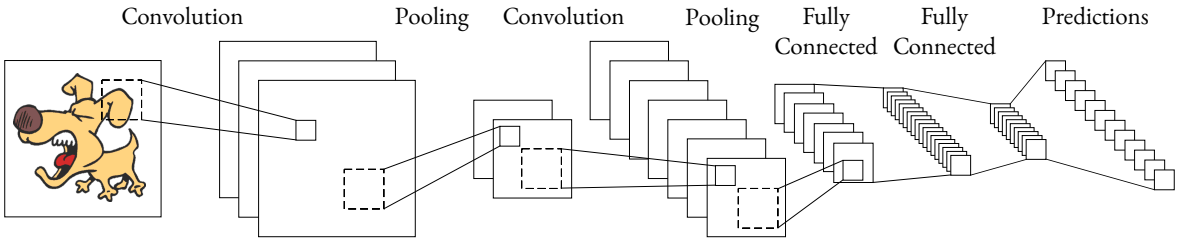


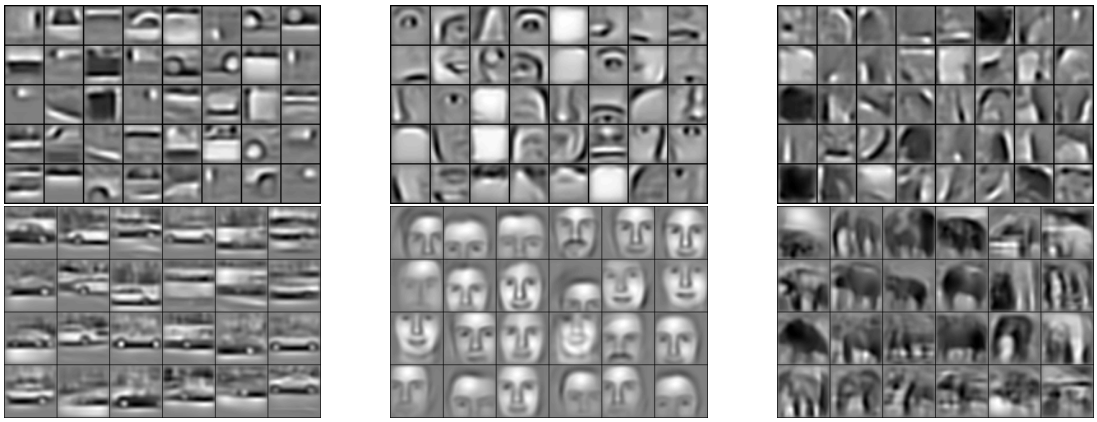
fig. 1.3: A illustration of a simple ConvNet architecture for image classification.

A convolutional layer is a linear transformation that preserves spatial information between pixels in the input image. A 2D convolutional layer is arranged in 3 dimensions: width, height and depth

with considering depth as channels, e.g. the colour images have three channels (RGB), while the grey-scaled images only have one.[2] In the hidden layers, the depth represents the number of feature maps. A pooling layer is further applied to reduce the dimensionality of feature maps while retaining the spatial information. We normally apply max-pooling which is simply taking the largest element from the feature map within a defined spatial patch.

To use a convolutional network for classification, a fully-connected layer is applied by passing the output of the model y with K dimensionality through an element-wise softmax function[3] to obtain a normalised probability score: $e^{y_j} / \sum_{i=1}^{K} e^{y_i}$ for each class j in total K classes.

ConvNets can learn hierarchical object-part representations in an unsupervised setting. Building on the first layer representations learned from the input images, the upper layer learned features by combining the previous layer's part representations into more complex, higher-level features as presented in Figure 1.4. Such properties ensure the model's robustness on high-dimensional, complex data and is the key to the prominent performance in many vision-based applications.



Car *face* *elephant*

fig. 1.4: Visualisation of the feature maps in the second layer (top) and the third layer (bottom) for various object categories using convolutional deep belief networks [LGRN09].

[2] A convolutional layer can be easily extended by adding more dimensions: e.g., a 3D convolutional layer is commonly used in video analysis which can preserve spatial-temporal features.

[3] Softmax literally means a soft version of argmax which represents a categorical distribution over K possible outcomes represented by a K dimensional vector of real values in the range of $(0, 1)$ that add up to 1.

1.2.2 Recurrent Neural Network

In the previous sections, we introduced several feed-forward neural networks. However, these networks depend on discrete data assuming no dependencies with fixed input length. For the interests of dealing with sequence data with various magnitude such as texts and sounds, we introduce recurrent neural networks (RNNs) which are able to not only learn the local and long-term temporal dependencies in the data but also accommodate the input sequences of variable length.

The major difference for RNNs is they *recurrently* use the same information to perform a task for every element of a sequence. Such properties make RNNs particularly good in language models such as machine translation and text generation.

A traditional RNN is composed by $x^{(t)}$: the input data x at time step t, $h^{(t)}$: the hidden state h at time step t, and $y^{(t)}$: the output at time step t. For each time step t, we compute

$$h^{(t)} = \phi(Ux^{(t)} + Wh^{(t-1)} + b_W)$$
$$y^{(t)} = \phi(Vh^{(t)} + b_y)$$

in which U, W, V are weights, b_W, b_y are the bias neurons and $\phi(\cdot)$ is a non-linear activation function. A RNN can be unfolded in time to become a feed-forward neural network. The general structure can be illustrated as follows.

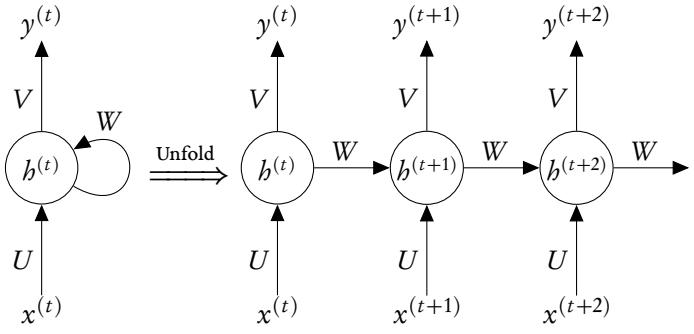


fig. 1.5: A illustration of unfolding a recurrent neural network into a feed-forward neural network.

Training a RNN is to use back-propagation through time (BPTT) to compute gradients. The chain rule however stacks many partial derivatives and makes RNN terrible at learning long-range

dependencies. Let's assume the error term of output neuron at time step n is $E^{(n)}$, we have

$$\frac{\partial E^{(n)}}{\partial W} = \sum_{i=1}^{n} \frac{\partial E^{(n)}}{\partial \hat{y}^{(n)}} \frac{\partial \hat{y}^{(n)}}{h^{(n)}} \left(\prod_{j=i+1}^{n} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(i)}}{\partial W}, \quad \hat{y}^{(n)} = \text{SoftMax}(V h^{(n)}),$$

where $\hat{y}_n$ is a prediction from the network.

Depending on our activation functions and network parameters, we could easily get exploding or vanishing gradients. [PMB13] discussed this problem and provided an effective solution by clipping the gradients at a threshold. The more popular solution is to use a Long Short-Term Memory (LSTM) [HS97] or Gated Recurrent Unit (GRU)[4] [CGCB14], some special kinds of RNNs that avoid the long-term dependency problem. [JZS15] further explored different types of recurrent network architectures and gave a thorough comparison.

1.3 Multi-Task Learning and Beyond

Deep neural networks have seen great success in a range of tasks, from image recognition [KSH12, GEB16] and machine translation [BCB14] to game playing [MKS+15, SHM+16] and cancer detection [CGGS13, EKN+17]. However, these networks are typically designed to achieve only one particular task. For building a more practical learning system in real-world applications, a network which can perform multiple tasks simultaneously is far more desirable than building a set of independent networks, one for each task. In the following sections, we will introduce the central theme of our research interests: multi-task learning and its related fields.

1.3.1 Multi-Task & Transfer Learning

Multi-Task Learning (MTL) [Car98] lies in a big umbrella of transfer learning. It is described as *inductive transfer learning* which aims at finding *good* feature representations to minimise domain divergence and classification or regression model error [PY10]. In order to fully characterise MTL,

[4] GRU is relatively new and computational more efficient, yet retaining performance on par with LSTM.

we refer to the definition from [ZY17]: "Given m learning tasks $\mathcal{T}_{i=1}^{m}$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for task $\mathcal{T}_i$ by using the knowledge contained in the m tasks."

In this thesis, we mainly discuss MTL under deep learning methods. One research aspect in MTL is multi-task framework design (Chapter 2). Compared to standard single-task learning, training multiple tasks whilst successfully learning a shared representation poses two key challenges:

i) **Loss Function (how to balance tasks)**: A multi-task loss function, which weights the relative contributions of each task, should enable learning of all tasks with equal importance, without allowing learning to be dominated by the easier tasks. Manual tuning of loss weights is tedious and sub-optimal, and so automatically learning these weights, or designing a network which is robust to these weights, is highly desirable. (Further discussions in Section 2.1.)

ii) **Network Architecture (how to share)**: A multi-task learning architecture should express both *task-agonistic* and *task-specific* features. As such, the network is encouraged to learn a generalisable representation (to avoid over-fitting), whilst also providing the ability to learn features tailored to each task (to avoid under-fitting). (Further discussions in Section 2.2.)

Multi-task learning has also often implicitly been used without explicit reference. The most common example is that of a neural network pre-trained with a large-scale dataset, such as ImageNet [KSH12], as a rich prior to a supervised fine-tuning [EBC+10] procedure. More explicitly, multi-task learning is often used with CNNs in computer vision to model two or even more related tasks jointly, such as image classification in multiple visual domains [RBV17], or to couple dense prediction tasks such as the estimation of depth maps, surface normals and semantic segmentation [MSGH16, EF15], or for pose estimation and action recognition [GHGM14].

Most multi-task learning network architectures are designed based on existing feed-forward deep neural networks. Some recent work includes [DZ17] which proposed a multi-task network based on ResNet-101 architecture [WJQ+17], with a lasso-regularised combination of features from different layers, to encourage the network to separate features that are useful for different tasks. The

Cross-stitch Network [MSGH16] used 'cross-stitch units' to combine two task-specific features, using a learnable linear combination of input activation maps. UberNet [Kok17] proposed a common CNN trunk which based on VGG-Net [SZ14], to perform as many as 7 tasks. The Progressive Networks method [RRD+16] applied a teacher-student relationship to accelerate training in multiple Atari games. [TBC+17] proposed *Distral*, a general framework for distilling common behaviours in multi-task reinforcement learning which show the resulting algorithms learn quicker and produce better performances with more stable and robust to hyper-parameter settings. However, some of the proposed multi-task networks are not parameter-efficient, since the network size increases linearly with the number of tasks. A desirable characteristic of multi-task learning is efficiency and this requires the network size to grow only gradually as extra tasks are added.

1.3.2 Auxiliary Learning

Apart from learning multiple tasks at once, in some situations, we only care about the performance of one or several (but not all) particular tasks (Chapter 3). Towards a better understanding on task relatedness, we designed experiments to understand the effect on generalisation by building auxiliary tasks with various attributes such as an increasing number of semantic classes or an increasing number of tasks. (Further discussions in Section 3.1.)

Auxiliary learning was not clearly defined, whilst there should be a common consensus that auxiliary tasks are designed to *assist* in finding a rich and robust representation, from which the ultimately desired main tasks profit. To give a formal description of auxiliary learning, we extend the definition of multi-task learning from [ZY17]: "Given m learning tasks $\mathcal{T}_{i=1}^{m}$ where all the tasks or a subset of them are related, auxiliary learning aims to help improve the learning of a model for task $\mathcal{T}_{j=1}^{n}, n < m$ by using the knowledge contained in the m tasks."

Though previously lacking the formal definition, researchers have shown the benefits by adding auxiliary tasks in the regime of multi-task learning in many applications. [TTLL17] applied auxiliary supervision with phoneme recognition at intermediate low-level representation of deep net-

works to improve the performance of conversational speech recognition. [LK18] chose auxiliary tasks which can be obtained with low effort, e.g., global descriptions of the scene to boost the performance for single scene depth estimation and semantic segmentation. [JMC+16] introduced a method for largely improving the learning and final performance of agents in Atari games as well as a 3D environment called Labyrinth by building auxiliary tasks to predict the onset of immediate rewards from a short historical context.

By forcing the network to generalise with more learning tasks with a fixed network capacity, auxiliary learning restricts the parameter space during optimisation and thus can be regarded as regularisation. However, by design, auxiliary tasks scale in different complexity and relatedness with the primary tasks from different applications. In Section 3.1, we will give a deep analysis on generalisation and network expressibility in auxiliary learning.

1.3.3 Meta Learning

Not only in the most common supervised learning we discussed above, MTL can also be easily combined with other learning paradigms including reinforcement learning, meta learning, active learning and even graphical models to improve the performance of learning tasks [ZY17]. Allowing multi-task models to determine *'what to share'* and *'how to share'* may temporarily circumvent the lack of theory and perform better to share knowledge even among loosely-related tasks.

Meta learning (or learning to learn) is a learning paradigm to optimise a neural network to better accomplish a task. In the context of AI systems, meta learning can be defined as the ability to acquire *knowledge versatility*. The literature on meta learning can be divided into several categories, including metric-based, optimisation-based and fully generic models. In the context of MTL, we mainly discuss optimisation-based meta learning.

Since optimisation for most modern deep neural networks relies on gradient descent, it is naturally to incorporate such learning framework into training meta learners. [RL16] proposed a LSTM-based meta learner which uses its state to represent the learning updates of the parameters of a clas-

sifiers and initial weights. [FAL17] proposed a simplified model that only learns the initial weights which is model-agonistic. [ADG+16] trained an LSTM to control parameter-update given each parameter's gradient information. This method was further improved with scalability and generalisation via a hierarchical recurrent architecture by [WMH+17]. In Section 3.2, we will discuss how to bring together with meta and auxiliary learning and design a general learning framework which can generate auxiliary knowledge automatically to improve generalisation in a multi-task network.

1.4 Applications

Multi-task learning has many applications across a variety of areas. Some of the works including in computer vision, speech, natural language processing and game playing have already been covered in the previous sections.

Other than common learning-based applications, MTL also helps to improve real-life applications such as in bioinformatics and health informatics. [WTAR10] improved MHC-I binding prediction and splice-site prediction with two multi-task multi-kernel methods. [WZY+12] proposed a sparse Bayesian model which learns correlations between features for all tasks to predict cognitive outcomes for the Alzheimers disease. [ZLZ+16] showed that the feature representation learned from large-scale nature images can transfer useful information to limited biological images and improve prediction accuracy.

The recent successes in MTL based on neural networks usually require a large number of training samples. By ingenious design of complimentary learning tasks, such constraints might be alleviated to perform *unsupervised* learning. For example, [FNPS16, ZBSL17] proposed image synthesis networks that generate new views by selecting pixels from nearby images. The relative pose of multiple cameras is used to predict the appearance of a nearby image such that the network can perform monocular depth estimation without ground truth depth knowledge at training time. Such property has unarguably raised another benefit from multi-task learning. And this self-supervised learning framework can be easily embedded in a robotic vision system for practical use.

1.5 Research Questions and Contributions

In this thesis, we aim to achieve a better understanding in MTL - the task relatedness, hierarchy, benefit as well as the connections to different learning paradigms and applications. We formulate the research questions and contributions in this work as follows.

- Research Question 1: *How to properly balance different types of tasks such that training multi-task networks will not be dominated by the easier task(s)?*

  In Section 2.1.2.2 and [LJD18] we present a novel weighting scheme, Dynamic Weight Average (DWA), which adapts the task weighting over time by considering the rate of change of the loss for each task. DWA does not touch the internal gradients in the network whilst it only requires the loss computed from each task to renormalise the gradient. Such design makes the implementation significantly simpler compared to other exiting weighting methods and yet still produces a noticeable performance boost over equal weighting method.

- Research Question 2: *How to build a multi-task learning architecture which is easy to train, parameter-efficient and robust to task weighting?*

  In Section 2.2.3 and [LJD18] we present a unified approach by designing a novel multi-task network based on the recent advances in attention mechanisms which (i) enables both task-shared and task-specific features to be learned automatically, and consequently (ii) learns an inherent robustness to the choice of loss weighting scheme.

  Our approach, the Multi-Task Attention Network (MTAN), consists of a single shared network containing a global feature pool, together with task-specific soft-attention modules. The flexibility from the attention enables much more expressive combinations of features to be learned for generalisation across tasks, whilst still allowing for discriminative features to be tailored for each individual task. Furthermore, automatically choosing which features to share and which to be task-specific allows for a highly efficient architecture, with far fewer parameters than multi-task architectures which have explicit separation of tasks.

- Research Question 3: *How to make use of multi-task relationships and design suitable auxiliary tasks to improve generalisation over primary tasks?*

  In Section 3.1 we perform multi-task learning both in single (image classification) and multiple visual domains (semantic segmentation + depth estimation). We build dataset with a multiple level of hierarchy (as similar to WordNet) and extensively analyse how different task complexity such as increasing number of auxiliary tasks or semantic classes can affect generalisation.

  We further propose two hypothesis to explain the generalisation gap and we confirm our observations to be robust under different of network architectures or optimisers.

- Research Question 4: *How to build a learning system such can automatically generate useful auxiliary tasks?*

  In Section 3.2 we introduce gradient descent meta learning based approach on auxiliary learning, which we call Meta AuXiliary Learning (MAXL) which is shown to generate auxiliary tasks automatically and can achieve similar or superior performance improvement compared with the human-designed tasks. Our approach is consisted with two components: a multi-task evaluator which is designed to learn primary and auxiliary tasks simultaneously and a meta learner which is designed to generate auxiliary knowledge used in the multi-task evaluator. The biggest strength of our approach lies in the splitting of data into training and meta-training data together with the meta learning that is performed via a second-derivative trick.

Much of the works in this thesis appears in the following publications:

- Shikun Liu, Edward Johns, and Andrew J. Davison, "End-to-End Multi-Task Learning with Attention," *ArXiv Preprint*, 2018.

- Shikun Liu, Edward Johns, and Andrew J. Davison, "Rethinking Generalisation with Auxiliary Learning," *ArXiv Preprint*, 2018.

2 Multi-Task Framework Design

In this chapter, we discuss recent advances in multi-task framework design. We first introduce adaptive weighting methods for dealing with task balancing problem in Section 2.1 in which includes our own contribution in Section 2.1.2.2. Furthermore, we introduce deep multi-task neural network design in Section 2.2 and our design which is based on attention in Section 2.2.3.

2.1 Adaptive Task Weighting

In general multi-task learning with K tasks, a loss function with input x and task-specific labels $y_i, i = 1, 2, \cdots, K$, is defined as

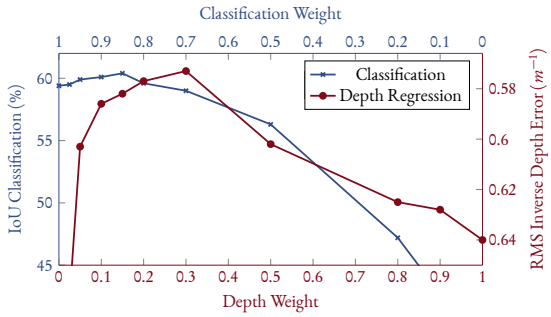
$$\mathscr{L}_{tot}(x, y_{1:K}) = \sum_{i=1}^{K} \lambda_i \mathscr{L}_i(x, y_i).$$

This is the linear combination of task-specific losses $\mathscr{L}_i$ with task weightings λ_i.

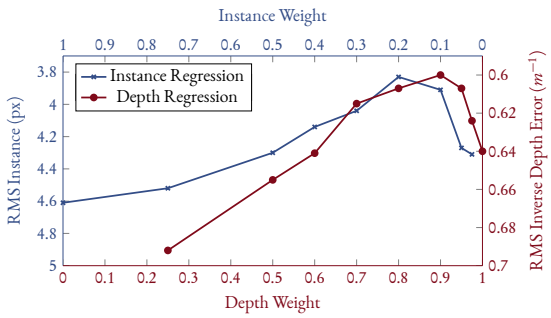
For many multi-task networks, training multiple tasks is difficult without finding the correct balance between those tasks. Different tasks which vary from the scale and complexity naturally produce different learning efficiency. For example, a task with low data variances would readily dominate the shared representations by producing a larger gradient magnitude. In an extreme case, we can roughly consider the network is learning task A with weight initialisation learned from task B.

A further supportive argument is provided in Figure 2.1 which shows a different combination of task weighting results in a noticeable performance change in each task[1].

[1] Applying a different combination of task weighting is equivalent to directly modifying gradient propagation.



Semantic Classification & Depth Regression



Instance Regression & Depth Regression

fig. 2.1: Comparing loss with multiple weighting combinations for selected two tasks in CityScapes dataset [COR+16]. Performance of the model in individual tasks is seen at both edges of the plot where $\lambda = 0$ or 1. Figures created by adaptations from [KGC18].

Recent approaches have attempted to address this issue [KGC18, LBBH98] by directly modifying multi-task loss function with Bayesian uncertainty (Section 2.1.1) or proposing an adaptive method to renormalise gradients such that task weighting λ_i can vary at each training step $t : \lambda_i = \lambda_i(t)$ (Section 2.1.2). We elaborate and explain these methods in the following sections.

2.1.1 Homoscedastic Uncertainty

Introduced in [KGC18], *Homoscedastic (or task-dependent)* uncertainty is a quantity varying between different tasks while stays constant for input data. In a multi-task setting, task uncertainty captures the relative confidence between tasks depending on the tasks' representation. It can be used as a measurement for weighting losses in a multi-task learning problem.

We suppose $f_\omega(x)$ be the output of a neural network with weight ω on input data x and ground truth prediction y. In the case of multiple model outputs with K discrete regression tasks, we obtain the following multi-task likelihood,

$$p(y_1, y_2, \cdots, y_k | f_\omega(x)) = \prod_{i=1}^{K} p(y_i | f_\omega(x)) \quad \text{and} \quad p(y_i | f_\omega(x)) = \mathcal{N}(f_\omega(x), \sigma_i^2)$$

where each model follows a Gaussian distribution with a noise scalar σ.

In maximum likelihood inference, we minimise the negative log likelihood[2] of the model which can be written as,

$$-\log p(y_1, y_2, \cdots, y_k | f_\omega(x)) \propto \sum_{i=1}^{K} \frac{1}{2\sigma_i^2} ||y_i - f_\omega(x)||^2 + \log \sigma_i .$$

We denote each task-specific loss $\mathscr{L}_i(\omega) = ||y_i - f_\omega(x)|^2$ and the final multi-task joint loss gives

$$\mathscr{L}_{tot}(\omega, \sigma_{1:K}) = \sum_{i=1}^{K} \frac{1}{2\sigma_i^2} \mathscr{L}_i(\omega) + \log \sigma_i .$$

The positive noise σ like the network parameter ω can be learned through back-propagation to denote the confidence in each task. As σ increases, we have the weight for the loss decreases. In practice, the authors suggested to predict the log variance $\log \sigma^2$ to avoid zero divisions and for the numerical stability.

One of the advantages for such method is that all parameters are learnable to minimise the cost of hyper-parameter search. However, though simplistic it seems, there are some other notable constrains in this design. First, our initial assumption[3] ensures that all K tasks are using the same network parameters ω, while our experiments (Section 2.3) show that such network is poorly designed which does not encourage to learn task-specific features. Thus, the method is not suitable for almost all network structures and feature sharing methods introduced in Section 2.2. Second, we found out that the performance is sensitive to different learning rates and the choice of gradient optimisation methods (also shown in Section 2.3).

2.1.2 Gradient Manipulation

Training deep neural network with back-propagation is to evaluate the gradient of loss function with respect to per-parameter in each layer, then the weights are updated with an optimisation rule. A rather straightforward approach on multi-task wight balancing problem is to directly manipulate

[2] In a classification network: $p(y|f_\omega(x)) = \text{SoftMax}(f_w(x))$ modelled with a softmax likelihood produces the same result. Detail derivations refer to the original paper.

[3] Also known as hard-parameter sharing in multi-task learning (detailed introduction in Section 2.2): to pick any feed-forward network and splits at the last layer for the prediction of each task.

network gradient in each training propagation step. In the follow sections, we investigate adaptive weighting problem in MTL with both implicit and explicit gradient-based weighting design.

2.1.2.1 Gradient Renormalisation

[CBLR17] offers a key insight in multi-task balancing problem, arguing that: "Task imbalances impede proper training because they manifest as imbalances between back-propagated gradients." It naturally follows that to balance each task is equivalent to balance gradient magnitudes computed from each task.

From this observation, the authors introduced *GradNorm*, an adaptive weighting method so that each task weighting λ_i can vary at each training step. GradNorm computes the gradient of the weighted single-task loss $\lambda_i \mathcal{L}_i$ in the last shared layer of weights $\omega' \subset \omega$ (subset of all network weights) and renormalise relative inverse training rate $r_i(t)$ to balance the task gradients $\lambda_i(t)$. The detailed algorithm is illustrated below.

Initialise: Task weightings: $\lambda_i(t=0) = 1, \forall i$
Initialise: Network parameters: ω
Initialise: Restoring force: α
for *each training time step* t **do**
    Compute: $\mathcal{L}(t) = \sum_{i=1}^{K} \lambda_i(t)\mathcal{L}_i(t)$ ◁ *Standard forward pass*
    Compute: $r_i(t) = \hat{\mathcal{L}}_i(t) / \mathbb{E}[\hat{\mathcal{L}}_i(t)]$ where $\hat{\mathcal{L}}_t(t) = \mathcal{L}_i(t)/\mathcal{L}(0)$
    Compute: $\hat{G}_\omega(t) = \mathbb{E}[G^{(i)}_{\omega'}(t)]$ where $G^{(i)}_{\omega'}(t) = ||\nabla_{\omega'} \lambda_i(t)\mathcal{L}_i(t)||_2$
    Compute: $\mathcal{L}_{grad} = \sum_{i=1}^{K} ||G_{\omega'}(t) - \hat{G}_{\omega'}(t) \times [r_i(t)]^\alpha||_1$
    Compute: $\nabla_{\lambda_i} \mathcal{L}_{grad}$ ◁ *GradNorm gradients*
    Compute: $\nabla_\omega \mathcal{L}(t)$ ◁ *Standard gradients*
    Update: $\lambda_i(t+1) \leftarrow \lambda_i(t)$ using $\nabla_{\lambda_i} \mathcal{L}_{grad}$
    Update: $\omega(t+1) \leftarrow \omega(t)$ using $\nabla_\omega \mathcal{L}(t)$ ◁ *Standard backward pass*
end

$\hat{G}_\omega(t)$ computes the average gradient norm across all tasks at training time and α represents the strength of restoring force, i.e., a higher value of α enforces stronger training rate balancing.

GradNorm like the weight uncertainty method introduced in the previous section is applied on the hard-parameter sharing in deep multi-task network. As we discussed previously, hard-parameter sharing is a ill-designed approach which does not encourage to learn task-specific representations. Besides, It also touches the internal network gradients which makes it rather difficult to implement in a modern deep learning framework.

2.1.2.2 Dynamic Weight Average (Our Approach)

Inspired by GradNorm [CBLR17], we propose a simple yet effective adaptive weighting method, named *Dynamic Weight Average* (DWA) which learns to average task weighting over time by considering the rate of change of loss for each task. But whilst GradNorm requires access to the network's internal gradients, our DWA proposal only requires the numerical task loss, and therefore its implementation is far simpler.

Our approach defines the task weighting λ_k for task k in two simple equations:

$$\lambda_k(t) := \frac{K \exp(w_k(t-1)/T)}{\sum_i \exp(w_i(t-1)/T)}, \quad w_k(t-1) = \frac{\mathscr{L}_k(t-1)}{\mathscr{L}_k(t-2)}$$

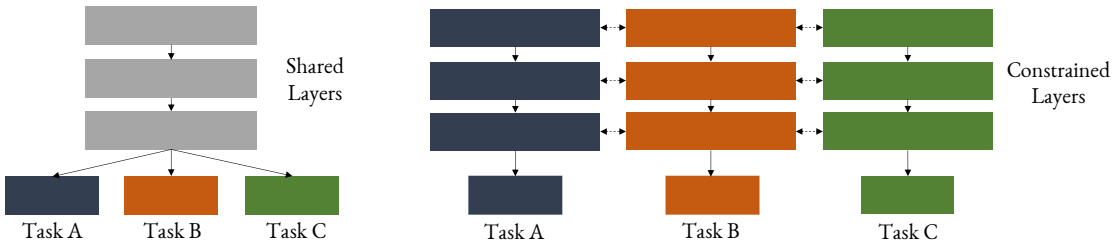
Here, $w_k(\cdot)$ calculates the relative descending rate in the range $(0, +\infty)$, t is an iteration index, and T represents a temperature which controls the softness of task weighting, similar to [HVD15] and α in GradNorm. A large T results in a more even distribution between different tasks. If T is large enough, we have $\lambda_i \approx 1$, and tasks are weighted equally. Finally, the softmax operator which is multiplied by K ensures that $\sum_i \lambda_i(t) = K$.

In our implementation, the loss value $\mathscr{L}_k(t)$ is calculated as the average loss in each epoch over several iterations. Doing so reduces the uncertainty from stochastic gradient descent and random training data selection. For $t = 1, 2$, we initialise $w_k(t) = 1$, but any non-balanced initialisation based on prior knowledge on training data could also be introduced.

2.2 Multi-Task Networks & Feature Sharing

Adaptive task weighings as shown in Section 2.3 and in [CBLR17, KGC18] are effective in certain types of multi-task architectures. However, it's always preferable to have a unified approach to (i) enable both task-shared and task-specific features to be learned automatically and (ii) learn an inherent robustness to the choice of loss weighting scheme.

In the context of deep learning, multi-task learning is typically done in *hard* or *soft-parameter sharing* of feature representations [Rud17]. Hard-parameter sharing greatly reduces over-fitting by using the same set of features across multiple learning tasks. On the other hand, soft-parameter sharing has more robust towards learning task-specific features where each task has its own model of parameters.



Hard-parameter Sharing *Soft-parameter Sharing*

fig. 2.2: Left: Hard-parameter sharing approach in which the input goes through the shared layers (grey) and splits at the last layer for task-specific predictions. Right: Soft-parameter sharing approach in which each task has its own feature parameters with certain constraints. Created by adaptations of original figures from [Rud17].

While most recent successes in deep multi-task network design are between in two sides: employ in a way to have both hard and soft-parameter sharing to incorporate task-shared as well as task-specific features. In the following sections, we introduce the most common varieties in the modern multi-task network design.

2.2.1 Fusion Network

Fusion network is a general feature learning technique which is being very popular in the recent proposals. The shared representations in the upper layers for each task is formed by learning task-specific coefficients to *fuse* lower feature representations from all tasks with linear combinations. In such way of design, each task will be supervised with useful features from related tasks but not the other.

Cross-Stitch Network [MSGH16] is one representative network which is built upon this feature fusion technique. It learns the optimal shared representations by cross-stitching two networks using linear combinations. Given two activation maps x_A, x_B from layer l for both tasks, we learn linear combinations $\tilde{x}_A, \tilde{x}_B$ by a cross-stitch operation parametrised by Λ. Specifically, at location (i, j) in the activation map, we perform

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \Lambda_{AA} & \Lambda_{AB} \\ \Lambda_{BA} & \Lambda_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix}.$$

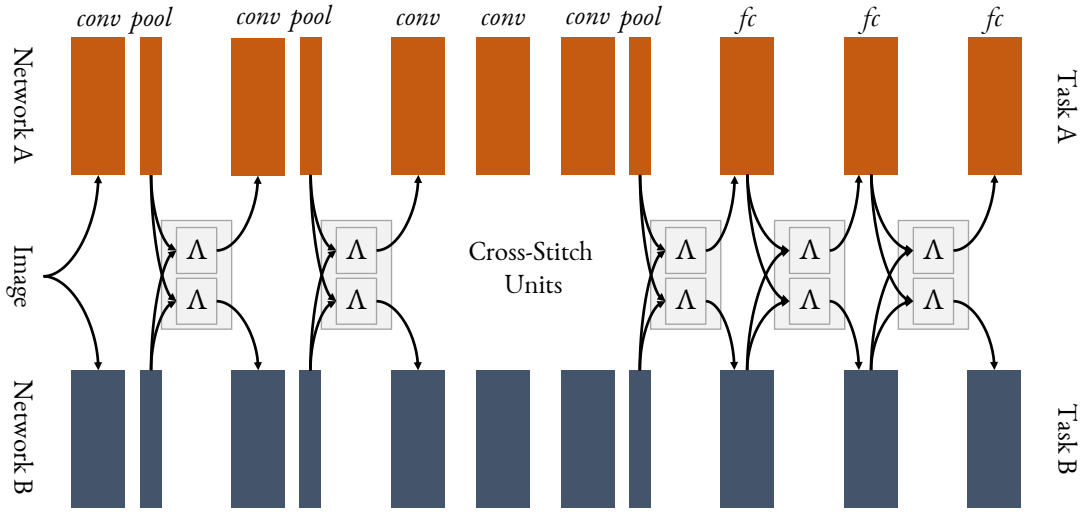


fig. 2.3: Using cross-stitch units to stitch two AlexNet [KSH12]. The cross-stitch units are only applied after pooling layers and fully-connected layers and thus can model shared representations as a linear combination of input activation maps. Created by adaptations of original figures from [MSGH16].

The optimal linear combinations are learnable through task-specific supervisions from a given set of tasks. Thus, the network can decide to make certain layers task specific by setting Λ_{AB} or Λ_{BA} to zero if two tasks are loosely related, or choose a more shared representation by assigning a higher value to them.

Nevertheless, cross-stitch network is computationally expensive since the network parameters grows linearly when we add more tasks. A more parameter-efficient feature fusion approach is when we apply linear combinations on one unified shared network.

In [DZ17], authors proposed a multi-task network built on ResNet-101 [HZRS16a] with lasso architecture, in which the representations that fed into each task head[4] is a sum of layer activations of residual units by learnable task-specific coefficients. The linear combinations are applied in full 23 candidate layers in block 3 of ResNet-101 architecture. Mathematically, the authors create a matrix Λ with K rows and M columns, where K is the number of tasks and M is the number of residual units in block 3. The representation passed to k^{th} task head is then,

$$\sum_{m=1}^{M} \Lambda_{k,m} \odot u_m,$$

where u_m is the output of residual unit m.

The authors further enforce that $\sum_{m=1}^{M} \Lambda_{k,m}^2 = 1$ for all task k, to control the output variance[5]. To encourage sparsity, they also add an $\mathscr{L}_1$ penalty on the entries of Λ to the final joint multi-task objective function.

This particular approach is extremely parameter efficient which only requires additional parameters from the Λ matrix and K task heads. Similar to the cross-stitch network, both of these feature fusion methods can be built on any type of feed-forward neural networks and thus can be trained in an end-to-end manner.

[4] In this work, authors perform tasks with a large varieties. The task head represents the task-specific representations after linear combinations. One need further apply dense layers for classification or decoder layers for pixel-wise regression, or other form of layers depending on the tasks.

[5] The entries in Λ can be negative, so a simple sum is insufficient.

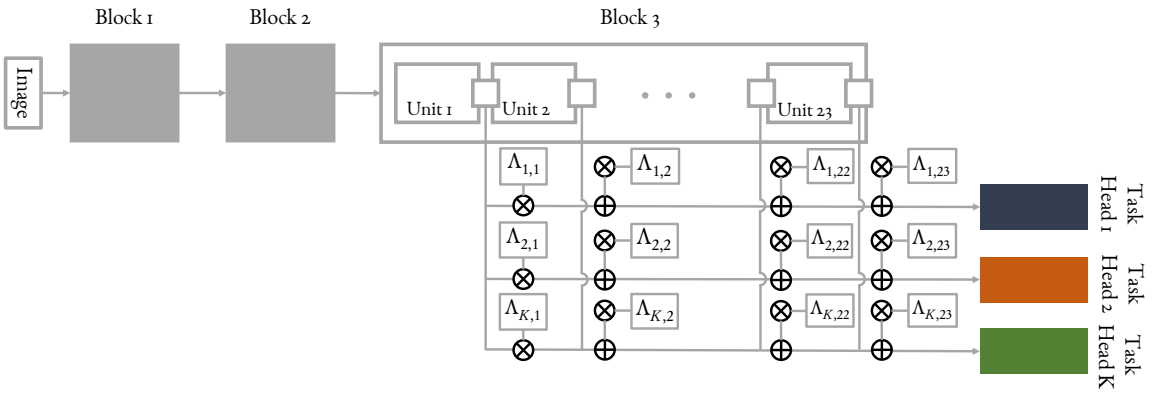


fig. 2.4: The lasso architecture based on ResNet-101. Each task head receives a linear combination of unit outputs within block 3. Created by adaptations of original figures from [DZ17].

2.2.2 Progressive Network

Introduced in [RRD+16], *Progressive Network* is an architecture with explicit support for transfer knowledge across sequences of tasks. Though it is more related in the field of transfer learning, it eventually solves K independent tasks at the end of training thus also fits in MTL.

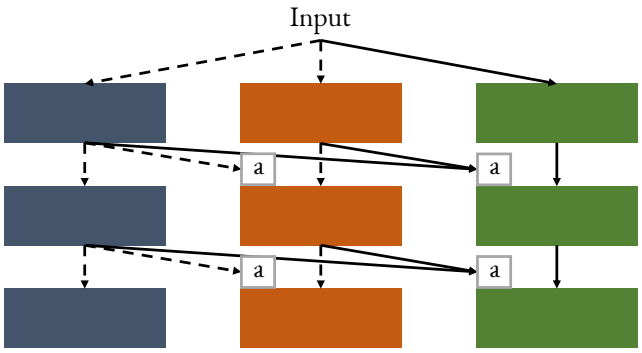


fig. 2.5: The structure of a three column progressive network. The first two columns on the left (dashed arrows) were trained on task 1 and 2 respectively. The third task (green) accesses to all previously learned features. Created by adaptations of original figures from [RRD+16].

The progressive network method applied a teacher-student relationship via lateral connections to accelerate training. While most networks train from scratch, progressive networks retain a pool of pre-trained models and accumulate previous learned knowledge via *adaptors* integrated at each layer of the feature hierarchy.

Some of key benefits include (i) immunity to catastrophic forgetting[6] (*freeze* the parameters in the source tasks) and (ii) accelerate training via prior learned knowledge. Furthermore, adaptors a ensures the number of parameters from the lateral connections to remain in the same order when more tasks are added.

However, this design also introduces new issues. For example, the authors make no assumptions on the task relationships, while the final performance naturally relies on the training order. To find the optimal training order for K tasks simply requires $K!$ experiments which is impossible to test out all the permutations in reality. Besides, the design is not able to train end-to-end and requires K independent network parameters as similar to cross-stitch network which is not efficient.

2.2.3 Attention Network (Our Approach)

We now introduce our novel multi-task learning architecture based on attention, the Multi-Task Attention Network (MTAN). The proposed architecture can readily be incorporated into any feed-forward network, and in the following we demonstrate how to build MTAN upon an encoder-decoder network, SegNet [BKC17]. This example configuration allows for image-to-image dense pixel-level prediction, such as semantic segmentation and depth prediction.

Architecture Design. MTAN consists of two components: a single shared network, and K task-specific attention networks. The shared network can be designed based on the particular task (in our case, SegNet, for image-to-image predictions), whilst each task-specific network consists of a set of attention modules, which link with the shared network. The attention modules apply a soft attention mask to the shared network, to determine the importance of each feature for the particular task. As such, the soft attention masks can be considered as feature selectors from the shared network, which are automatically learned in an end-to-end manner, whilst the shared network learns a compact global pool of features across all tasks.

[6] Catastrophic forgetting is a phenomenon particularly in neural networks when the networks completely forget previous learned information when learning a new task.

A detailed visualisation of our network based on VGG-16 [SZ14] is shown in Figure 2.6, which displays the encoder half of SegNet. The decoder half of SegNet is then symmetric to VGG-16. As shown, each attention module learns a soft attention mask, which itself is dependent on the features in the shared network at the corresponding layer. Therefore, the features in the shared network, and the soft attention masks, can be learned jointly to maximise the generalisation of the shared features across multiple tasks, whilst simultaneously maximising the task-specific performance due to the soft attention.

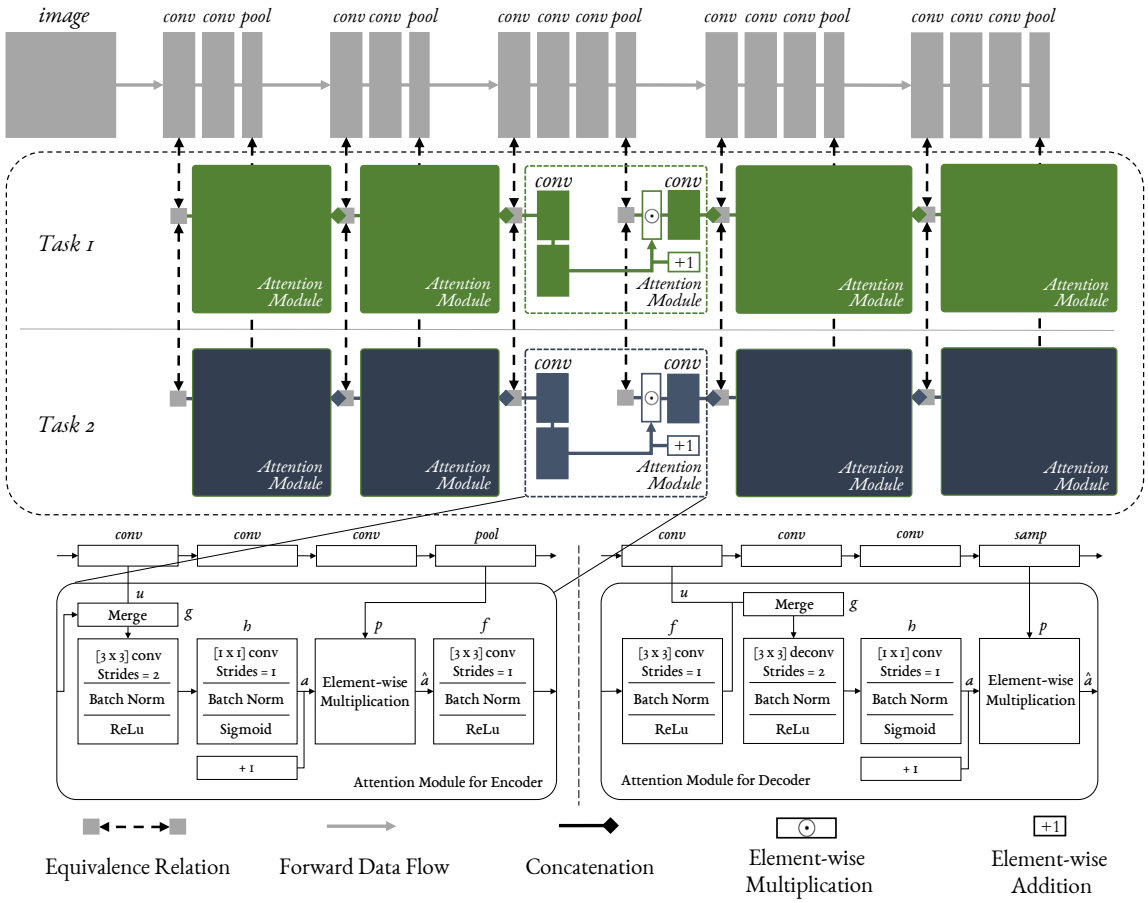


fig. 2.6: Visualisation of MTAN based on VGG-16, showing the encoder half of SegNet (with the decoder half being symmetrical to the encoder). Task one (green) and task two (blue) have their own set of attention modules, which link with the shared network (grey). Here, the middle of the five attention modules has its structure exposed for visualisation, which is then further expanded in the bottom section of the figure, showing both the encoder and decoder versions of the module.

Task Specific Attention Module. The attention module is designed to allow the task-specific network to learn task-related features, by applying a soft attention mask to the features in the shared network, with one attention mask per task per feature channel. We denote the shared features in the j^{th} block of the shared network as $p^{(j)}$, and the learned attention mask in this layer for task i as $a_i^{(j)}$. The task-specific features $\hat{a}_i^{(j)}$ in this layer, are then computed by element-wise multiplication of the attention masks with the shared features:

$$\hat{a}_i^{(j)} = (a_i^{(j)} + 1) \odot p^{(j)},$$

where $\odot$ denotes element-wise multiplication. The '+1' operation is the residual identity mapping motivated by [WJQ+17, HZRS16b], to help the network learn more robust attention maps by avoiding exploding or vanishing gradients, which may otherwise be caused by consecutive layer-by-layer multiplications.

As shown in Figure 2.6, for the first attention module in the encoder, the attended features only take input features in the shared network. But for subsequent attention modules in layer $j \geq 2$, the input is formed by a concatenation of the shared features $u^{(j)}$, and the task-specific features from the previous layer $\hat{a}_i^{(j-1)}$:

$$a_i^{(j)} = h\left(g\left(\left[u^{(j)}; f\left(\hat{a}_i^{(j-1)}\right)\right]\right)\right), \quad j \geq 2,$$

Here, f, g, h are convolutional (or deconvolutional) layers with batch normalisation, following a non-linear activation ReLu in f, g or Sigmoid in h. Both f and g are composed with a $[3 \times 3]$ kernel, while h uses a $[1 \times 1]$ kernel to match the channels between the concatenated features and the shared features. Furthermore, in function g, we apply a stride of size 2, to match the compressed / up-sampled resolution from the pooling / up-sampling operation. See Figure 2.6 for the equivalence of architecture between the encoder and decoder.

The attention mask $a_i^{(j)} \in [0, 1]$ is learned in a self-supervised fashion with back-propagation. If $a_i^{(j)} \to 0$, the attended feature maps are equivalent to global feature maps and the tasks share all the features. Therefore, we expect the performance to be no worse than that of a shared multi-task network, which splits into individual tasks only at the end of the network, and we show results demonstrating this in Section 2.3.

2.3 Experimental Results

In this section, we introduce the dataset used for validation in Section 2.3.1, several baselines for comparison to our MTAN method in Section 2.3.2 and learning tasks for evaluation in Section 2.3.3. In Section 2.3.4, we show the effectiveness of MTAN with various weighting methods compared with single and multi-task baseline methods. To further understand the attention function, we present the visualisation of the learned attention masks in Section 2.3.5.

2.3.1 Datasets

CityScapes. The CityScapes dataset [COR+16] includes 2975 training and 500 validation high-resolution images, with publicly-available annotations. We use this dataset for two tasks: semantic segmentation and depth estimation. To speed up training, all training and validation images were resized to $[128 \times 256]$ resolution. The dataset contains 19 classes from 7 categories for pixel-wise semantic segmentation, together with pixel-wise depth labels. We pair the depth estimation task with the coarser 7-class defined in the original CityScapes dataset.

NYUv2. The NYUv2 dataset [NSF12] includes roughly 800 RGB-D images for both training and validation set. We evaluate our method on semantic class sets with 13 labels described in [CFNL13]. The depth data for this dataset is recorded by depth cameras from Microsoft Kinect, and the surface normal are computed with the method from [ZP+14]. To speed up training, all training and validation images were resized to $[288 \times 384]$ resolution.

To test the robustness of our multi-task network, we perform experiments both in outdoor dataset CityScapes and indoor dataset NYUv2 . Road scene images from CityScapes have limited variation with a standard spatial arrangements which can be easily captured by a deep network. The difficulty in outdoor scenes relies on small objects such as pedestrians and vehicles of far sight. In comparison, images of indoor scenes are much more complex since the view points can vary a lot and the appearance for objects in the same categories widely vary in texture and shape as well as in different

lighting conditions. The pixel-wise dense prediction in indoor scene remains one of the hardest challenges for architectures and methods design in computer vision.

2.3.2 Baselines

Most image-to-image multi-task learning architectures are designed based on specific feed-forward neural networks, and thus they are usually not directly comparable. Therefore, for a fair comparison across multi-task learning methods, we designed 4 networks (1 single-task + 3 multi-task) based on SegNet [PMB13], which we consider as baselines:

- **STAN-SegNet:** Single-Task Attention Network, where we directly apply our proposed MTAN whilst only performing a single task.

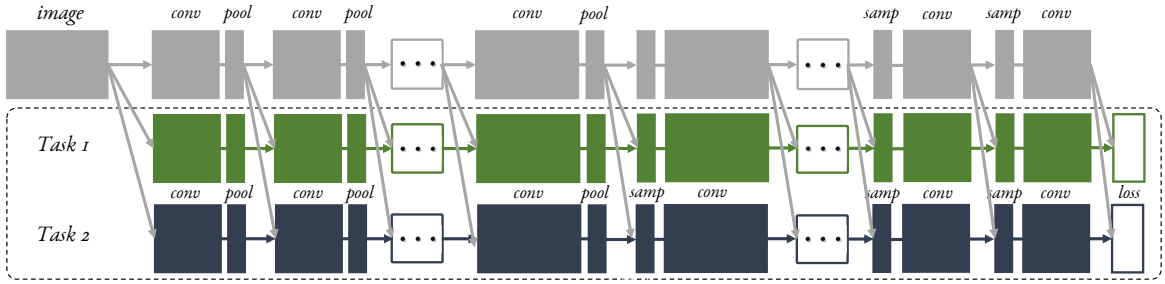
- **SegNet, Split:** The vanilla feedforward SegNet, which splits at the last layer for the final prediction of two tasks.

- **MTLBL1-SegNet:** A shared network with two task-specific networks, where each task-specific network receives all features from the shared network, without any attention module. This is similar to the cross-stitch network [MSGH16], but replaces the cross-stitches operations with an additional shared network for a closer comparison to our method[7].

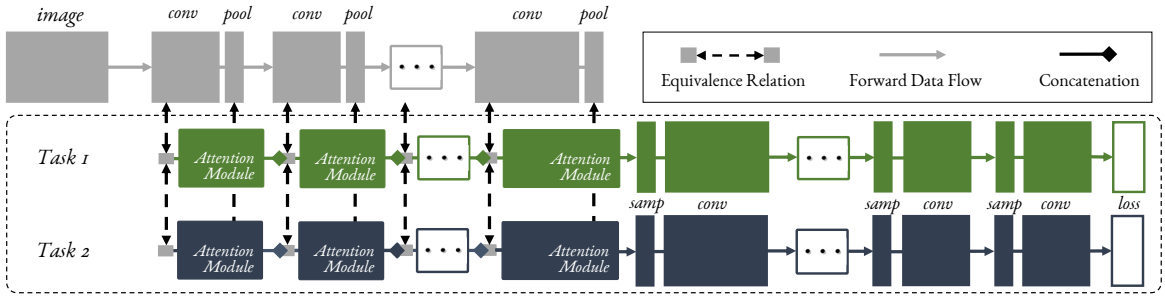
- **MTLBL2-SegNet:** The encoder part of the network is the same as our proposed MTAN-SegNet, but we don't apply attention modules to the decoder. Each task-specific decoder network takes its only task-specific features and uses the standard SegNet decoder to return task predictions.

Both baselines MTLBL1-SegNet and MTLBL2-SegNet have more parameters than our proposed MTAN-SegNet, and were tested to validate that our proposed method's better performance is due to the attention modules, rather than simply due to the increase in network parameters. A detailed visualisation of these two baselines is presented in Figure 2.7.

[7]Note that the original Cross-Stitch Network scales poorly with the number of tasks, whereas our method scales sub-linearly.



(a) Multi-task Learning Baseline 1



(b) Multi-task Learning Baseline 2

fig. 2.7: Two proposed Multi-task Learning Baseline based on SegNet.

2.3.3 Learning Tasks

In our application to image-to-image pixel-level dense prediction, we perform the following three tasks, where $\hat{\mathrm{Y}}$ represents the network predicted result, and Y represents the ground-truth label:

Semantic Segmentation. For semantic segmentation, we apply a pixel-wise cross-entropy for each predicted class label from a depth-softmax classifier. We average the result for each valid pixel.

$$\mathscr{L}_1(\mathrm{X},\mathrm{Y}_1) = -\frac{1}{pq}\sum_{p,q}\mathrm{Y}_1(p,q)\log\hat{\mathrm{Y}}_1(p,q).$$

Depth Estimation. For depth estimation, we apply an L_1 norm comparing the inverse predicted and ground-truth depth, as inverse depth can more easily represent points at infinite distances (such as the sky). The depth data for CityScape was calculated using the SGM algorithm [Hir08], and is represented as inverse ground-truth depth. In NYUv2, the inverse ground truth depth was calcu-

lated by $2/(D+2)$, where D is the original depth map from the dataset.

$$\mathscr{L}_2(\mathrm{X},\mathrm{Y}_2)=\frac{1}{pq}\sum_{p,q}|\mathrm{Y}_2(p,q)-\hat{\mathrm{Y}}_2(p,q)|.$$

Surface Normals. For surface normal (only in NYUv2 dataset), we normalise the vector at each pixel to unit L_2 norm and employ an element-wise loss at each pixel using a dot product.

$$\mathscr{L}_3(\mathrm{X},\mathrm{Y}_3)=-\frac{1}{pq}\sum_{p,q}\mathrm{Y}_3(p,q)\cdot\hat{\mathrm{Y}}_3(p,q).$$

2.3.4 Results on Multi-Task Learning

We now evaluate the performance of our proposed MTAN method in image-to-image multi-task learning, based on the SegNet architecture. Using the CityScapes and NYUv2 dataset described in Section 2.3.1, we compare all the baseline methods introduced in the previous section, together with vanilla SegNet itself [BKC17].

Training. For each network architecture, we ran experiments with three types of weighting methods: equal weighting, weight uncertainty [KGC18], and our proposed DWA (with temperature $T = 20$ and 5, found empirically to be optimum for CityScapes and NYUv2 respectively). We trained all the models with stochastic gradient descent using a learning rate of 0.01 and momentum of 0.9, with a batch size of 8. During training, we divide the learning rate by 2 for every 50 epochs, for a total of 150 epochs (except for the weight uncertainty method which drops the learning rate after every 25 epochs). During training, we discovered that the weight uncertainty method [KGC18] is not robust to learning rate. Therefore, to compare fairly to this method, we drop the learning rate significantly to avoid premature saturation of the gradients.

Results. Table 2.1 shows experimental results across all architectures, and all multi task weighting schemes for both CityScapes and NYUv2 dataset. By comparing validation results, our method either outperforms all other methods, across all weighting schemes, and across both tasks or performs at least as well as our proposed baselines, despite they have a significant larger number of parameter space.

Table 2.1: Training and validation results for CityScapes (up) and NYUv2 dataset (down). We measure segmentation as the mean intersection-over-union (mIoU) for CityScapes / accuracy (Acc.) for NYUv2 (the higher the better), depth estimation as relative absolute error (RAE) for CityScapes / mean absolute error (MAE) for NYUv2 (the lower the better) and surface normal as cosine similarity $(1 + Cos.)$ (the lower the better). Column #P compares the number of network parameters, and the best performing combination of multi-task architecture and weighting is highlighted in bold with top validation scores for each task are annotated with boxes.

Type	#P.	Architecture	Weighting	Semantic (mIoU)		Depth (RAE)	
				Train	Val	Train	Val
Single Task	1	Vanilla SegNet [BKC17]	n.a.	0.7012	0.5097	0.3101	0.5027
	1.32	STAN - SegNet	n.a.	0.6950	0.5200	0.4535	0.5535
Multi Task	≈1	Vanilla SegNet, Split	Equal Weights	0.6850	0.5067	0.3542	0.5666
			Uncert. Weights [KGC18]	**0.6965**	0.4957	0.4434	0.5726
			DWA, $T = 20$	0.6937	**0.5112**	**0.3540**	**0.4969**
	3.02	MTLBL1-SegNet	Equal Weights	0.7104	0.5164	0.4066	0.5166
			Uncert. Weights [KGC18]	**0.7459**	0.5126	0.4480	0.5352
			DWA, $T = 20$	0.7126	**0.5202**	**0.3333**	**0.4491**
	2.06	MTLBL2-SegNet	Equal Weights	0.6672	0.5161	0.3817	0.5019
			Uncert. Weights [KGC18]	**0.6833**	0.5130	0.4478	0.5440
			DWA, $T = 20$	0.6736	**0.5188**	**0.3714**	**0.4786**
	1.64	MTAN-SegNet	Equal Weights	0.6929	**0.5268**	0.3490	0.4246
			Uncert. Weights [KGC18]	**0.6957**	0.5152	0.4318	0.4768
			DWA, $T = 20$	0.6863	0.5259	**0.3417**	**0.4184**

Type	#P.	Architecture	Weighting	Semantic (Acc.)		Depth (MAE)		Normal (1+Cos.)	
				Train	Val	Train	Val	Train	Val
Single Task	1	Vanilla SegNet	n.a.	0.8508	0.4521	0.0213	0.0717	0.0078	0.0577
	1.32	STAN - SegNet	n.a.	0.9784	0.5181	0.0194	0.0665	0.0096	0.0568
Multi Task	≈ 1	Vanilla SegNet, Split	Equal	**0.8619**	0.4668	0.0327	0.0692	0.0317	0.0570
			Uncertainty	0.8552	0.4708	**0.0285**	0.0676	**0.0203**	0.0575
			DWA, $T = 5$	0.8556	**0.4943**	0.0387	**0.0624**	0.0355	**0.0502**
	4.03	MTLBL1-SegNet	Equal	0.9902	0.5196	0.0157	0.0630	0.0053	0.0522
			Uncertainty	0.9903	**0.5344**	**0.0148**	0.0622	**0.0039**	0.0516
			DWA, $T = 5$	**0.9921**	0.5328	0.0155	**0.0609**	0.0050	**0.0506**
	3.09	MTLBL2-SegNet	Equal Weights	0.9901	0.5032	0.0142	0.0650	0.0067	0.0568
			Uncertainty	0.9906	0.5129	**0.0130**	**0.0607**	**0.0059**	0.0556
			DWA, $T = 5$	**0.9920**	**0.5271**	0.0136	0.0636	0.0063	**0.0549**
	1.96	MTAN-SegNet	Equal	0.9666	**0.5321**	**0.0275**	**0.0610**	0.0222	**0.0492**
			Uncertainty	**0.9778**	0.4918	0.0328	0.0618	**0.0070**	0.0507
			DWA, $T = 5$	0.9614	0.5012	0.0308	0.0678	0.0246	0.0503

Moreover, our method has two key advantages. First, due to the efficiency of having a single shared feature pool with attention masks automatically learning which features to share, our method outperforms other methods without requiring extra parameters (column #P), and even with significantly fewer parameters in some cases. Second, our method is more robust to the choice of weighting scheme than other methods, and does not require cumbersome tweaking of loss weights. We can also see that our proposed DWA weighting method performs best across most of the baselines, whereas the uncertainty method [KGC18] appears to overfit by displaying good training performance, but poorer validation performance.

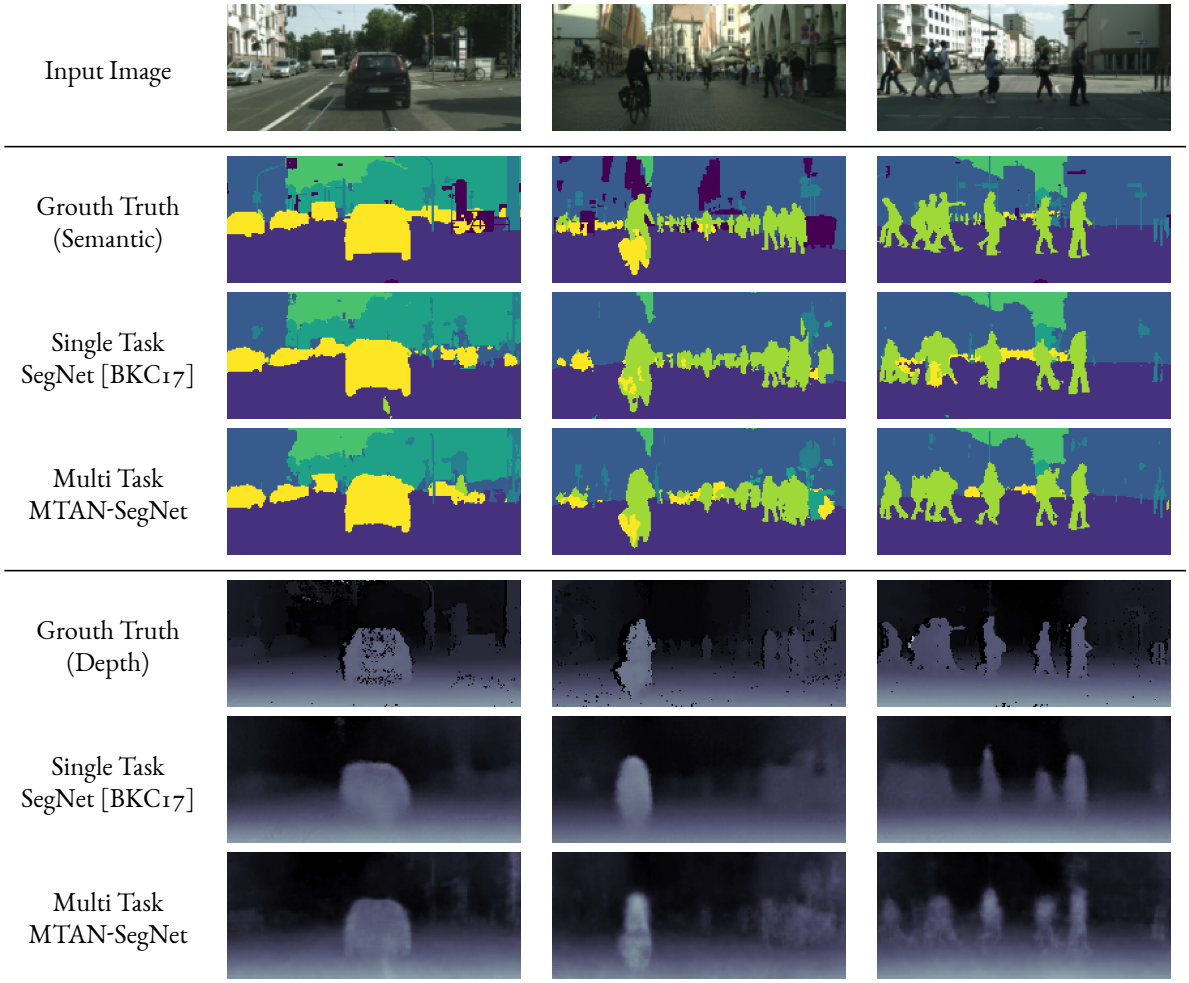
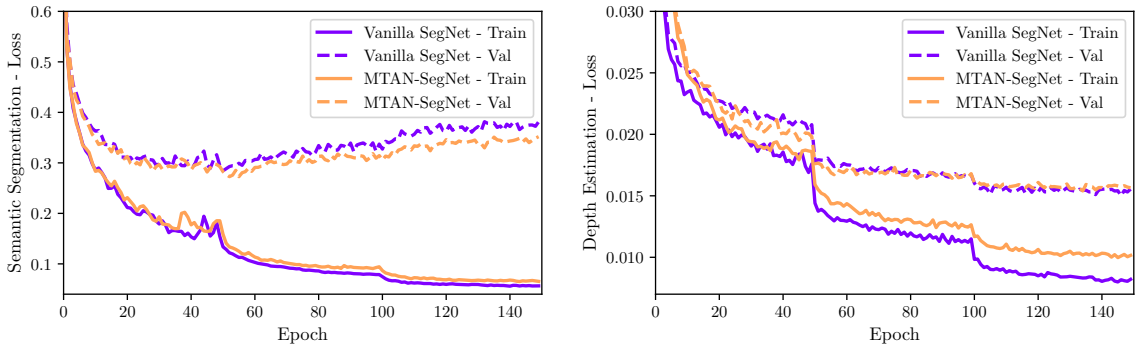


fig. 2.8: CityScapes dataset validation results on 7-class semantic labelling and depth estimation all trained with equal weighting. The original images are cropped to avoid invalid points for better visualisation.

To further investigate generalisation performance of our method compared to the single task SegNet, in Figure 2.9 we plot the learning curve with respect to the loss of both tasks. We can clearly see that our network is able to alleviate overfitting (reduces the gap between training and validation) compared to single task training, and produces a better generalisable feature representation. Figure 2.8 then shows qualitative results and comparison. We can see the advantage of multi-task learning particularly for depth estimation, where the edges of objects are clearly more pronounced compared with single-task training.



Semantic Loss Curve *Depth Loss Curve*

fig. 2.9: Learning curves of semantic and depth loss between vanilla SegNet [BKC17] and our network MTAN-SegNet.

2.3.5 Attention Masks as Feature Selectors

Finally, to further understand the role of the proposed attention modules, in Figure 2.10 we visualise the layer 1 attention masks learned with our network. We can see a clear difference in attention masks between the two tasks, with each mask working as a feature selector to mask out uninformative parts of the shared features, and focus on parts which are either task-specific, or task-shared. In particular, the attention masks have strong similarity to the shared features, and thus appear to act as a feature augmentation, whereas the attention maps for depth estimation appear to act as sparse feature extractors.

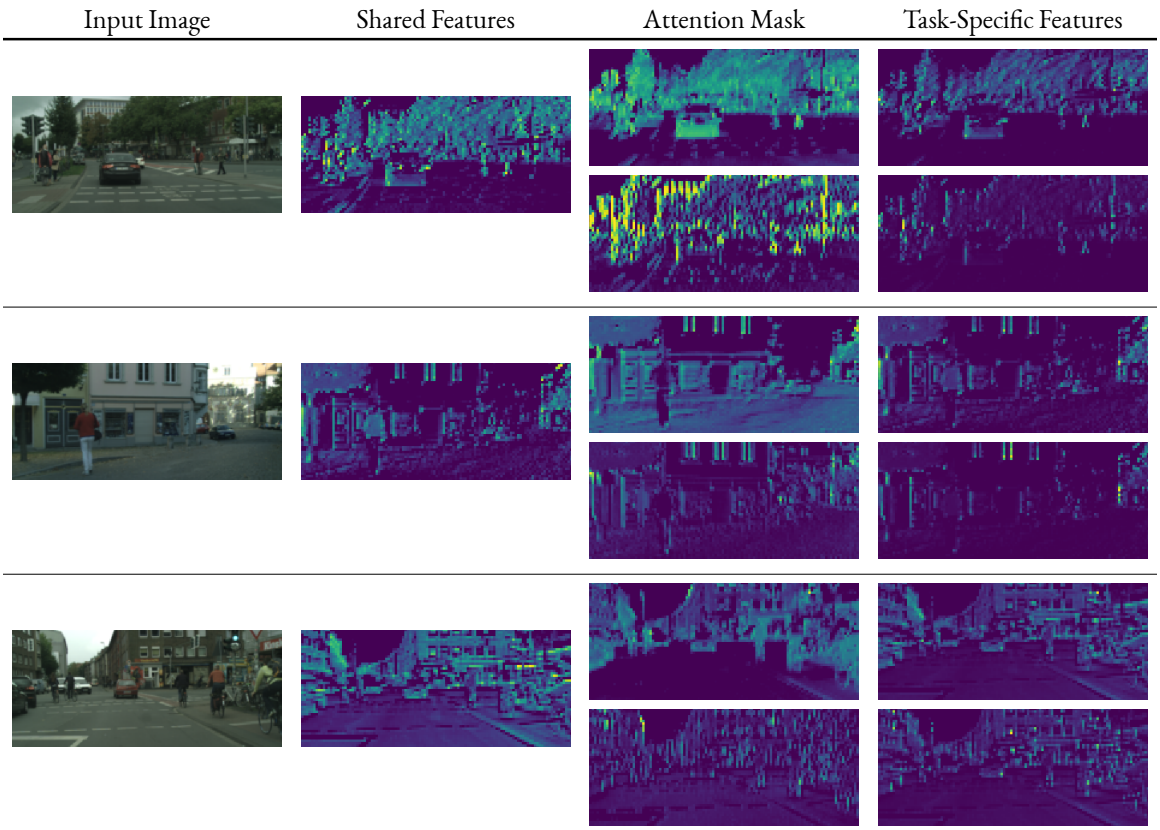


fig. 2.10: Visualisation of the first layer of 7-class semantic segmentation and depth estimation attention features of our proposed network. Top row: semantic features; Bottom row: depth features. The colours for each image are rescaled to fit the data.

Learning with Auxiliary Tasks 3

In some learning scenarios, we only need to perform well on a subset of all learning tasks. In this chapter, we introduce auxiliary learning, a new type of learning paradigm which focuses on building auxiliary tasks particularly to improve generalisation over a chosen subset of learning tasks. We first perform a deep analysis on multi-task relationships in Section 3.1 and understand generalisation behaviours with different task attributes. We further propose a general meta learning framework in Section 3.2, which is able to generate auxiliary tasks automatically to improve generalisation and on top of that performs as well as human-designed tasks.

3.1 Generalisation & Expressibility in Auxiliary Learning

In this work, we first re-introspect generalisation in deep neural networks with an auxiliary learning setting in both single and multi domains. We train a multi-task network to simultaneously predict a set of learning tasks in which each auxiliary task is chosen as one level of hierarchical label from a defined multi-label dataset. In all learning tasks, one task is marked as the primary task, and all the rest learning tasks are marked as auxiliary tasks. Our goal is to understand how generalisation over the primary task behaves when we have auxiliary tasks with different class complexity.

In single domain, we measure class complexity in each learning task by the number of classes in the dataset, i.e., the finer classification label which describes with a more detailed information gives a higher complexity. Similarly, in multi domain, we measure complexity as the number of semantic classes in the segmentation map for each image.

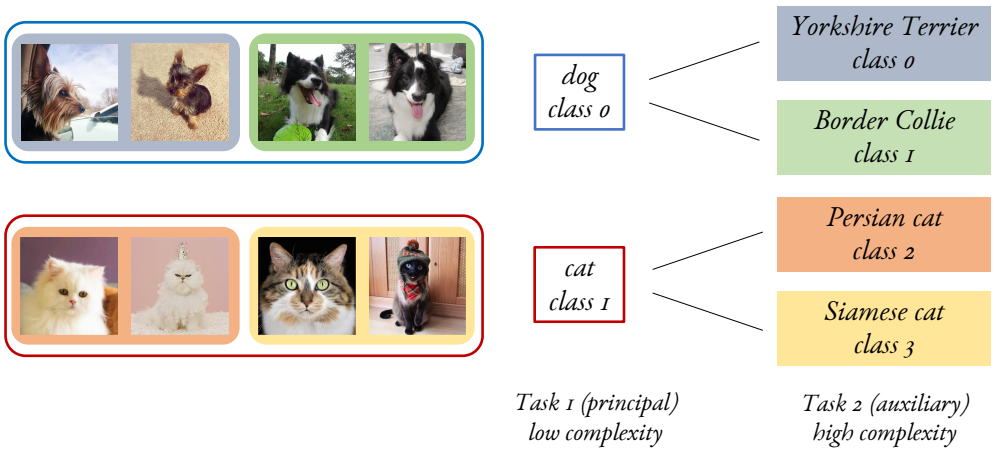


fig. 3.1: An illustration for our dataset in single domain in which every image has a hierarchical multi-label information. The courser label contains a lower complexity, whilst the finer label contains a higher complexity. Each colour corresponds to the label information of its image.

We show that the generalisation of primary task is radically different when we accompany auxiliary tasks with different class complexities. Our findings can be summarised as,

i) The primary task in auxiliary learning will have improved performance when we accompany auxiliary tasks with higher complexity.

ii) The performance of the primary task in auxiliary learning depends on the auxiliary task with the highest performance improvement (no matter how many auxiliary tasks given).

In the following sections, we perform experiments to support our findings. Then, we provide a new perspective on generalisation in the regime of deep learning to conclude the results from both single and multi domains.

3.1.1 Image Pixel-wise Prediction in Multi Domains

To understand the effect of class complexity in multi domain, we reuse the CityScapes dataset as presented in Section 2.3.1. We evaluate our network on training data with different number of semantic classes (as described in Table 3.1), leaving the depth labels the same across all experiments.

We trained the network with the same settings as in Section 2.3.4, but with all networks having equal loss weighting.

Table 3.1: Three levels of semantic classes for the CityScapes data used in our experiments.

2-class	7-class	19-class
background	void	void
	flat	road, sidewalk
	construction	building, wall, fence
	object	pole, traffic light, traffic sign
	nature	vegetation, terrain
	sky	sky
foreground	human	person, rider
	vehicle	carm truck, bus, caravan, trailer, train, motorcycle

We pair the depth estimation task with three levels of semantic segmentation using 2, 7 or 19 classes (excluding the void group in 7 and 19 classes). Labels for the 19 classes are the original ground-truth labels, and the coarser 7 categories are defined as in the original CityScapes dataset. We further create a 2-class dataset with only background and foreground object classes. The details of these segmentation classes are presented in Table 3.1. Please note that both the 7 and 19-class CityScapes datasets have a void class which is not used in network training.

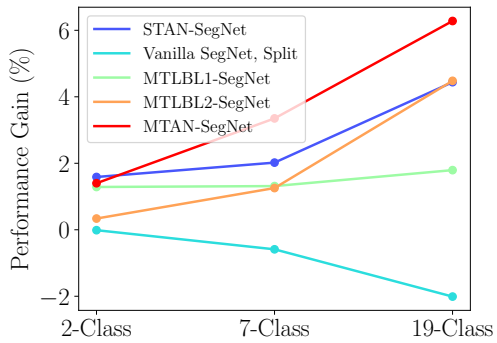
Training. We apply the same training pipeline described in Section 2.3.4 with our proposed multi-task attention network (Section 2.2.3) compared with baseline methods in Section 2.3.2.

Results. Table 3.2 shows validation results for these experiments. Note that in single task depth estimation, changing semantic class will have no effect so we leave this part of the table blank. In Figure 3.2, the performance gain of all multi-task methods (including our own), compared to the single-task SegNet [BKC17], is shown.

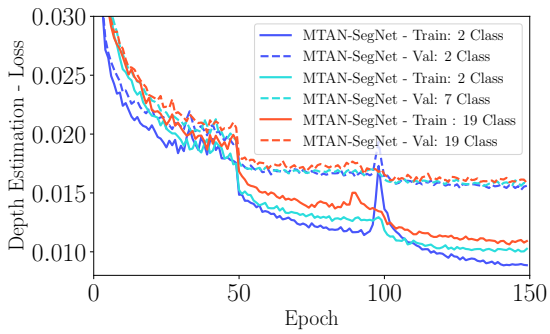
There are three interesting findings in this experiment. First, we observe that the multi-task performance gain (over the single task network) increases as the number of semantic classes increases. In fact, for only a 2-class setup, the single-task framework performs best. However, for greater

Table 3.2: 2/7/19-class semantic segmentation and depth estimation results trained with equal weighting on validation set of CityScapes dataset. We measure semantic segmentation as mean intersection-over-union (mIoU) (the higher the better) and depth estimation as relative absolute error (RAE) (the lower the better).

Type	Method	Semantic (mIoU)			Depth (RAE)		
		2-class	7-class	19-class	2-class	7-class	19-class
Single Task	Vanilla SegNet [BKC17]	0.8004	0.5097	0.2675	-	-	-
	STAN - SegNet	**0.8127**	0.5200	0.2794	-	-	-
Multi Task	Vanilla SegNet, Split	0.7999	0.5067	0.2621	**0.5289**	0.5666	0.4748
	MTLBL1-SegNet	0.8103	0.5164	0.2723	0.5419	0.5166	0.4893
	MTLBL2-SegNet	0.8027	0.5161	0.2795	0.6204	0.5019	0.4700
	MTAN-SegNet	0.8112	**0.5268**	**0.2843**	0.6067	**0.4246**	**0.4513**



Performance Gain



Depth Loss Curve

fig. 3.2: Left: Performance gain from all single-task and multi-task baseline methods compared with results produced SegNet. Right: Depth loss from training and validation set in MTAN.

task complexity, the multi-task framework encourages the sharing of features, for a more efficient use of available network parameters, which then leads to better results. Second, the complimentary depth estimation task performs better when it trains with more semantic classes, owing to the greater provision of complementary information from which shared features can be learned for generalisation, further supporting the benefit of multi-task learning. Third, Figure 3.2 then shows that for multi-task learning, greater task complexity causes the gap between validation loss and training loss to actually decrease, and hence overfitting to decrease, due to the ability to share features across tasks using the supervised labels.

As a deeper look into the performance change in MTAN-SegNet compared to single task SegNet, we further provide performance change in Figure 3.3 of our method compared to the single-task SegNet. Here, the categories are sorted in increasing order of the mean percentage image coverage. We can see our network outperform the single task baseline in 16 out of 19 classes. In particular, there are the greatest performance boosts in small to mid sizes of classes, whilst classes with large image coverage tend to already perform well with single-task learning. This further highlights the power of multi-task learning when laballed data is scarce, but suggests that when sufficient data is available, single-task learning can still be effective.

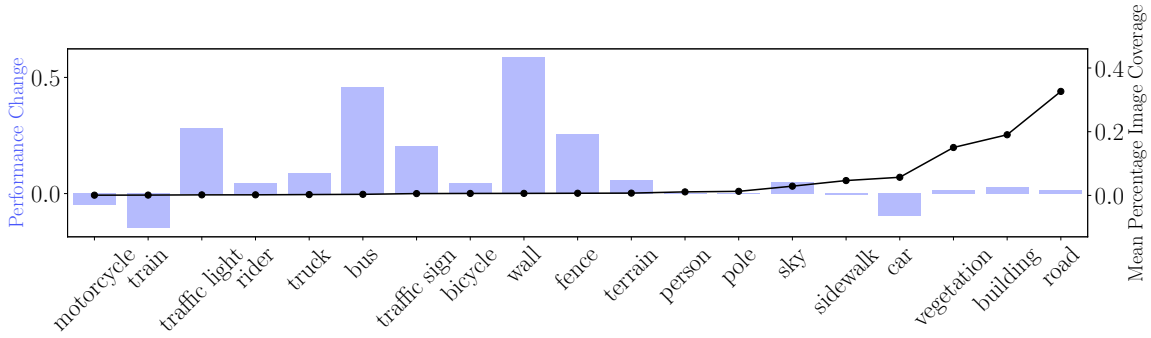


fig. 3.3: Performance comparison between MTAN-SegNet and SegNet trained in each 19 dense class. We report performance gain in bar chart (purple) sorted in mean percentage image coverage (black) for each class.

3.1.2 Object Classification in Single Domain

In single domain with object classification, we build a 4-level hierarchy (3-10-20-100) based on CIFAR100 dataset with human knowledge (see Table 3.3 for the full details). Considering auxiliary learning for object classification in a single domain setting, the shared multi-task network takes one single image and predicts one chosen level of our pre-defined hierarchical multi-label information in each classification task.

Training. We first perform experiments with one task chosen as 3/10/20-class label and a second task chosen to be 100-class label and we compare the test performance between auxiliary training and single task training (with no auxiliary tasks). We optimise the multi-task network with vanilla

Table 3.3: Building a 4-level hierarchy for image classification task based on CIFAR100 dataset.

3 Class	10 Class	20 Class	100 Class
animals	large animals	reptiles	crocodile, dinosaur, lizard, snake, turtle
		large carnivores	bear, leopard, lion, tiger, wolf
		large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
	medium animals	aquatic mammals	beaver, dolphin, otter, seal, whale
		medium-sized mammals	fox, porcupine, possum, raccoon, skunk
	small animals	small mammals	hamster, mouse, rabbit, shrew, squirrel
		fish	aquarium fish, flatfish, ray, shark, trout
	invertebrates	insects	bee, beetle, butterfly, caterpillar, cockroach
		non-insect invertebrates	crab, lobster, snail, spider, worm
	people	people	baby, boy, girl, man, woman
vegetations	vegetations	flowers	orchids, poppies, roses, sunflowers, tulips
		fruit and vegetables	apples, mushrooms, oranges, pears, peppers
		trees	maple, oak, palm, pine, willow
objects and scenes	household objects	food containers	bottles, bowls, cans, cups, plates
		household electrical devices	clock, keyboard, lamp, telephone, television
		household furniture	bed, chair, couch, table, wardrobe
	construction	large man-made outdoor things	bridge, castle, house, road, skyscraper
	natural scenes	large natural outdoor scenes	cloud, forest, mountain, plain, sea
	vechicles	vehicles 1	bicycle, bus, motorcycle, pickup truck, train
		vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

stochastic gradient descent with learning rate 0.01. During training, we drop the learning rate by half for every 50 epochs. For the fair comparison, we keep all our hyper-parameters the same across all the experiments. In each task, we multiply a constant $1/N$ for N number of classification tasks (e.g. $N = 2$ for training with one primary and one auxiliary task) to normalise the gradient such that the shared network receives a similar level of gradient from the combination of back-propagations from N tasks.

To make sure the generalisation behaviour of learning performance to be consistent and robust across different learning methods and network architectures, we train the network both with and without regularisation in VGG-16 [SZ14] and ResNet-50 [HZRS16a] respectively with a hard parameter sharing approach. We provide the test performance in Figure 3.4.

Results. In Figure 3.4, we observe that in either case, the performance of all tasks with 3/10/20-class improve dramatically when pairing with the auxiliary task with 100-class whilst the performance on the task with 100-class pairing with 3/10/20-class drops a certain degree of performance compared to single task training.

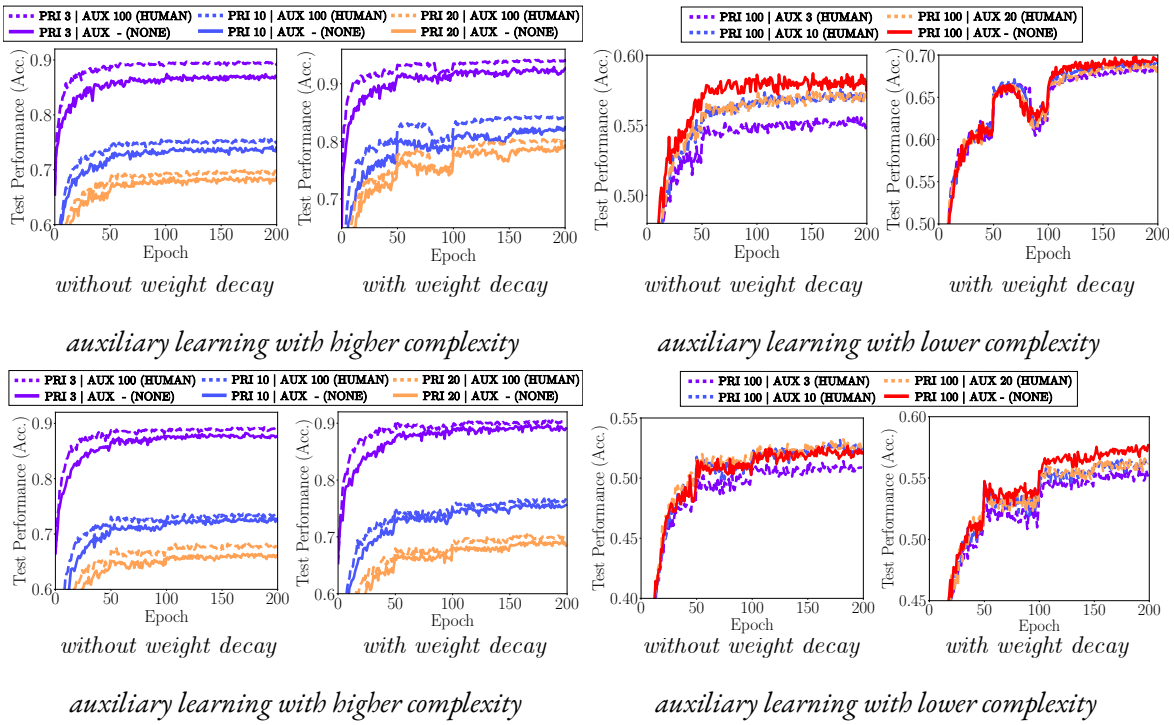


fig. 3.4: The test performance on the primary task in VGG-16 (up) and ResNet-50 (down) pairing with auxiliary tasks with a higher or lower class complexity.

However, what causes the primary task to achieve a better or worse generalisation remains unknown. Though there are possibly more underlying reasons, we propose two hypotheses to explain this generalisation gap,

- Hypothesis 1: The higher complexity from auxiliary task brings more finer information which assists the primary task to learn a more generalisable feature representation.

- Hypothesis 2: The higher complexity from auxiliary task is working as a regulariser which changes the gradient dynamics during training with back-propagation and thus avoid over-fitting.

To further explore whether generalisation in auxiliary learning lies in our two proposed hypotheses, we further perform experiments in which we consider 3-class as our primary task and a combination of a subset of 10/20/100-class as our auxiliary tasks. In Figure 3.5 left, we show that the performance of primary task will further improve when we have higher auxiliary class complexity. However, it will reach the highest improvement rate and decrease when the auxiliary class complexity is excessive (i.e., pairing with 100-class performs worse than 20-class). In Figure 3.5 right, we show that the performance of primary task only depends on the auxiliary task which provides the best performance (i.e., pairing 20-class alone is the same as pairing a set of auxiliary tasks containing 20-class).

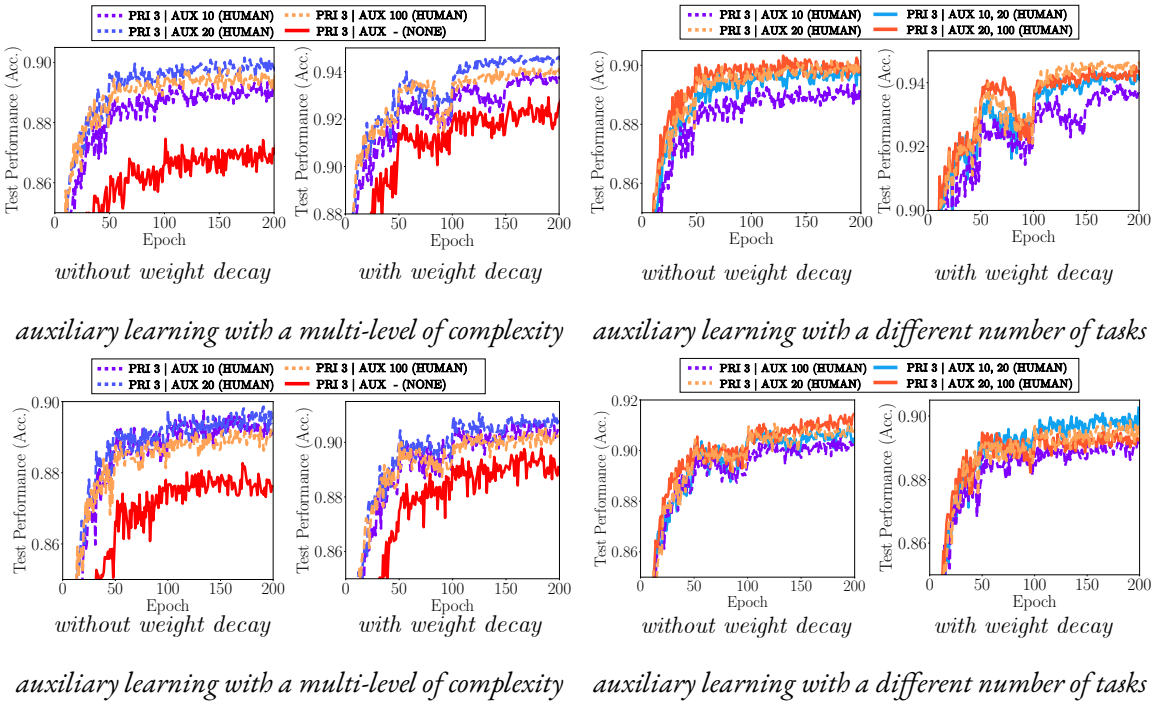


fig. 3.5: The test performance on the primary task in VGG-16 (up) and ResNet-50 (down) pairing with auxiliary tasks with a multi-level class complexity.

In Figure 3.5, we believe both hypotheses are true in auxiliary learning whilst there should be a tradeoff between the weighting of each hypothesis. In the most extreme case, when the auxiliary task has the highest complexity where each image is assigned as one class, we may consider auxiliary task contains no useful information and only works as a regulariser by back-propagating noise.

3.2 Meta Auxiliary Learning

In the previous section, we discussed the generalisation behaviour in an auxiliary learning setting. It is naturally to think about, is it possible to generate auxiliary tasks automatically and improve generalisation? In this section, we introduce a gradient descent meta learning based approach on auxiliary learning which we called Meta AuXiliary Learning (MAXL), is shown to generate higher complexity task only counter-intuitively based on the primary task with low complexity. In a sense, it learns to generate *knowledge* to assist the primary task by minimising domain divergence.

3.2.1 Problem Set-up & Model Objectives

The goal of meta auxiliary learning is to train a meta generator that can generate higher complexity task to improve performance of the primary task. To accomplish this, we have two networks: a multi-task evaluator (which is the same applied in Section 3.1) and a meta generator to generate auxiliary tasks used in the multi-task evaluator. For simplicity, we consider classification tasks in the following sections, whilst this meta approach can be easily generalised to any type of tasks.

We denote the multi-task evaluator as a function $f_{\theta_1}(x)$ that takes an input x with network parameters θ_1 and the meta generator as a function $g_{\theta_2}(x)$ that takes the same input x with network parameters θ_2. For each dataset, we split into three subsets: training $(x_{\text{train}}, y_{\text{train}})$, validation $(x_{\text{val}}, y_{\text{val}})$ and test $(x_{\text{test}}, y_{\text{test}})$ with input data x and its corresponding ground truth label y for the primary task. Particularly, training data are used for updating the θ_1 and validation data are used for updating the θ_2 and we use test data for performance evaluation.

In multi-task evaluator, we apply a hard parameter sharing approach in which we predict the primary and auxiliary tasks using the same set of features θ_1 in the multi-task network. At the end of the last feature layer, we then further apply task-specific layers to output the corresponding prediction for each task. We denote the predicted primary labels by $f^{\text{pri}}_{\theta_1}(x)$ and predicted auxiliary labels by $f^{\text{aux}}_{\theta_1}(x)$.

In meta generator, we have the same input x as for the multi-task evaluator and it outputs the last feature representation as $g_{\theta_2}(x)$. For image classification task, in order to produce the constrained predictions with a pre-defined hierarchical structure, we need the additional input ground truth primary labels y and a pre-defined hierarchical structure ψ which represents the number of sub-classes for each upper class. Having these additional inputs, we may generate the auxiliary labels with $g_{\theta_2}(x, y, \psi)$ via a mask (constrained) SoftMax for the final prediction. The detailed explanation of mask SoftMax is descibred in Section 3.2.2. The visualisation of the our proposed MAXL approach is shown in Figure 3.6.

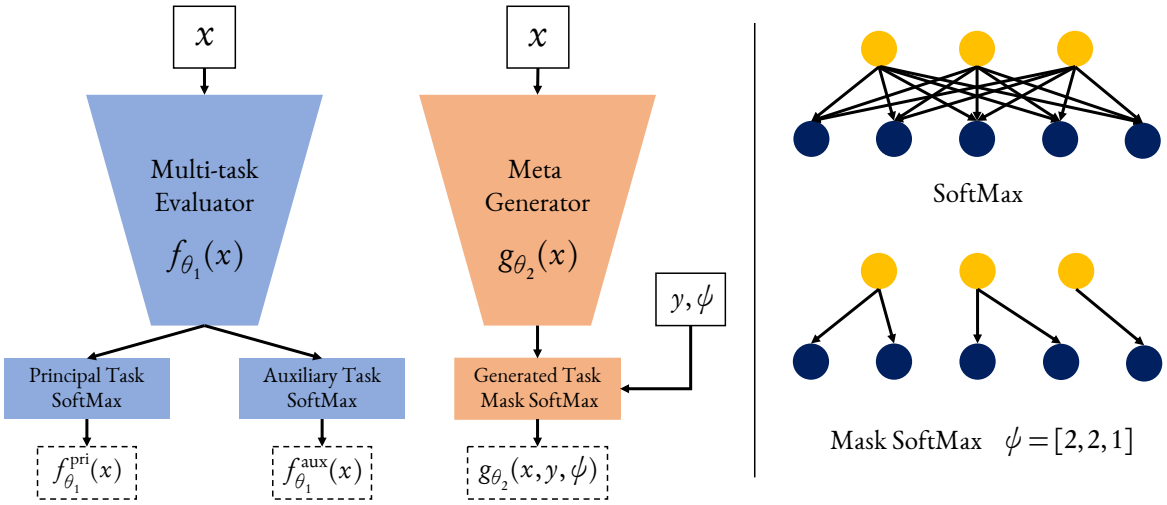


fig. 3.6: Left: Visualisation of two networks applied in meta auxiliary learning algorithm. Right: Visualisation of vanilla SoftMax and Mask SoftMax as for an example of primary 3-class and auxiliary 5-class task. Compared to SoftMax, Mask SoftMax has an additional hierarchical structure $\phi = [2, 2, 1]$ to constrain prediction space.

For both primary and auxiliary classification tasks, we apply focal loss [LGG+17] with a focusing parameter $\gamma = 2$ which is defined as,

$$\mathcal{L}(\hat{y}, y) = -y(1-\hat{y})^{\gamma} \log(\hat{y})$$

where $\hat{y}$ is the predicted label and y is the ground-truth label.

The focal loss which learns to focus on the incorrectly predicted labels can further improve the performance during our experimental evaluation compared with the regular cross entropy log loss.

To update the parameter space θ_1 in multi-task evaluator, we define the multi-task objective as follows:

$$\arg\min_{\theta_1}\Big(\mathscr{L}(f^{\text{pri}}_{\theta_1}(x^{(i)}_{\text{train}}), y^{(i)}_{\text{train}}) + \mathscr{L}(f^{\text{aux}}_{\theta_1}(x^{(i)}_{\text{train}}), g_{\theta_2}(x^{(i)}_{\text{train}}, y^{(i)}_{\text{train}}, \psi))\Big)$$

in which (i) is denoted as i^{th} batch from the training data.

Training the multi-task network first requires the generated auxiliary tasks produced from meta generator. It then takes a direct combination of losses computed from both primary and auxiliary tasks. For each back-propagation for this combined multi-task loss, the network is learning to predict the human defined primary task as well as and the generated auxiliary task.

To update the parameter space θ_2 in meta generator, we define the meta objective as follows,

$$\arg\min_{\theta_2}\mathscr{L}(f^{\text{pri}}_{\theta_1^+}(x^{(i)}_{\text{val}}), y^{(i)}_{\text{val}})$$

in which

$$\theta_1^+ = \theta_1 - \alpha\nabla_{\theta_1}\Big(\mathscr{L}(f^{\text{pri}}_{\theta_1}(x^{(i)}_{\text{val}}), y^{(i)}_{\text{val}}) + \mathscr{L}(f^{\text{aux}}_{\theta_1}(x^{(i)}_{\text{val}}), g_{\theta_2}(x^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}, \psi)\Big),$$

α is a hyper-parameter learning rate and the updated θ_1^+ holds the same structure of the multi-task loss used in updating the θ_1 after one step of gradient descent.

The trick in meta objective is that we perform the derivative over a derivative (a Hession matrix) to update the θ_2 by using a retained computing graph of θ_1^+ to connect the relationship with θ_2 without having a zero gradient. The meta objective is trying to generate the best auxiliary labels such that to maximise the validation performance of the primary task. This second derivative trick in meta learning was also appeared in [FAL17] and [LZCL17].

However, we found out that the generated auxiliary label can easily collapse (i.e. degenerate into the similar class complexity as to one of the primary task) that hits a local minima without producing any extra useful knowledge. Thus, to encourage the network learning more meaningful information, we further apply an entropy loss $\mathscr{H}(g_{\theta_2}(x^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}, \psi))$ as a regularisation term in the meta objective. The detailed explanation of the entropy loss and the collapsing label problem will be discussed in Section 3.2.3.

Lastly, the entire MAXL algorithm pipeline is defined as follows:

Dataset: $D = \left\{(x_{\text{train}}, y_{\text{train}}), (x_{\text{val}}, y_{\text{val}})\right\}$

Initialise: Network parameters: θ_1, θ_2; Sub-class number: ψ

Initialise: Hyper-parameter (learning rate): α, β; Hyper-parameter (task weighting): λ

for *each training iteration* i **do**

 # fetch one batch of training and validation data

$\left\{(x_{\text{train}}^{(i)}, y_{\text{train}}^{(i)}), (x_{\text{val}}^{(i)}, y_{\text{val}}^{(i)})\right\} \in \left\{(x_{\text{train}}, y_{\text{train}}), (x_{\text{val}}, y_{\text{val}})\right\}$

 # training step

Update: $\theta_1 \leftarrow \theta_1 - \alpha \nabla_{\theta_1} \left(\mathscr{L}(f_{\theta_1}^{\text{pri}}(x_{\text{train}}^{(i)}), y_{\text{train}}^{(i)}) + \mathscr{L}(f_{\theta_1}^{\text{aux}}(x_{\text{train}}^{(i)}), g_{\theta_2}(x_{\text{train}}^{(i)}, y_{\text{train}}^{(i)}, \psi)) \right)$

 # meta-training step

Compute: $\theta_1^+ = \theta_1 - \alpha \nabla_{\theta_1} \left(\mathscr{L}(f_{\theta_1}^{\text{pri}}(x_{\text{val}}^{(i)}), y_{\text{val}}^{(i)}) + \mathscr{L}(f_{\theta_1}^{\text{aux}}(x_{\text{val}}^{(i)}), g_{\theta_2}(x_{\text{val}}^{(i)}, y_{\text{val}}^{(i)}, \psi)) \right)$

Update: $\theta_2 \leftarrow \theta_2 - \beta \nabla_{\theta_2} \left(\mathscr{L}(f_{\theta_1^+}^{\text{pri}}(x_{\text{val}}^{(i)}), y_{\text{val}}^{(i)}) + \lambda \mathscr{H}(g_{\theta_2}(x_{\text{val}}^{(i)}, y_{\text{val}}^{(i)}, \psi)) \right)$

end

3.2.2 Mask SoftMax for Hierarchical Predictions

In the prediction layer of the meta generator, we design a modified SoftMax function to predict the correct hierarchical labels with a pre-defined hierarchy ψ.

As shown in Figure 3.6 (upper right), the original softmax function which does not depend on the upper hierarchy labels (in the primary task) could not predict the expected hierarchical labels without any prediction constrains. Thus, it basically performs (re)-clustering with a larger (thus finer) label space. While the mask SoftMax resolves this issue by multiplying a binary mask on the original SoftMax function.

Building the binary mask M depends on the primary ground-truth label y and a hierarchical structure ψ. ψ is defined to have the same number of the primary label in which each element $\psi[i]$ represents the i^{th} number of sub-classes in each upper-class and thus we have the total prediction space for auxiliary label is $\sum_i \psi[i]$.

We build the binary mask $M = \mathscr{B}(y, \psi)$ with a binarise function $\mathscr{B}$. The primary ground-truth label y first picks a range of corresponding hierarchical sub-classes $\psi[y]$ and then create a binary mask M with size $\sum_i \psi[i]$ in a multi one-hot encoding $\mathbb{1}_{\sum_{i<y}\psi[i]:\sum_{i<y+1}\psi[i]}$ ($\mathbb{1}_{a:b}$ is denoted as an one-hot encoding in which indexes from a to b are encoded as 1).

Let's again use the example in Figure 3.6. Considering the primary task to be class 3 with ground truth labels $y = 0, 1, 2$ and hierarchical structure $\psi = [2, 2, 1]$, we will have the auxiliary prediction space is equal to 5 and corresponding binary masks $M = [1, 1, 0, 0, 0], [0, 0, 1, 1, 0], [0, 0, 0, 0, 1]$ respectively.

Finally, we apply binary mask M with an element-wise multiplication on the original SoftMax function for the final hierarchical predictions,

$$\text{SoftMax:} \quad p(\hat{y}_i) = \frac{\exp \hat{y}_i}{\sum_i \exp \hat{y}_i}, \qquad \text{Mask SoftMax:} \quad p(\hat{y}_i) = \frac{\exp M \odot \hat{y}_i}{\sum_i \exp M \odot \hat{y}_i}, \quad M = \mathscr{B}(y, \psi).$$

in which $p(\hat{y}_i)$ represents the probability of the predicted primary label $\hat{y}$ over class i and $\odot$ represents element-wise multiplication.

3.2.3 The Collapsing Class Problem

As discussed in the previous sections, we predict each auxiliary label with a hierarchical structure ψ. However, the number of sub-classes defined in $\psi[i]$ is the maximal auxiliary label prediction space with no guarantee all $\psi[i]$ classes will be predicted. In another word, in some cases, some auxiliary labels defined in $\psi[i]$ can be overlooked and thus collapsed into a smaller class size unless/until for the case when $\psi[i] = 1$ (no hierarchy defined in upper class i). In experiments, we found out this phenomenon is particularly serious when we have a large learning rate for training meta generator or with a really large hierarchy ψ.

To avoid the collapsing class problem, in addition to applying a smaller learning rate, we also design an additional regularisation loss, which we called the entropy loss $\mathscr{H}(\hat{y}^{(i)})$ to encourage the meta generator utilise the entire prediction space by maximising prediction entropy.

Assuming we have a well-balanced dataset, the entropy loss is trying to calculate the KL distance between the predicted auxiliary label space $\hat{y}^{(i)}$ and a uniform distribution $\mathcal{U}$ for each i^{th} batch. It is equivalent to calculate the entropy of the predicted label space and is defined as the follows,

$$\mathcal{H}(\hat{y}^{(i)}) = \sum_{k=1}^{K} \overline{y_k} \log \overline{y_k}, \quad \overline{y_k} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}^{(i)}.$$

in which K is the number of prediction label size and N is the training batch size.

In a well-balanced dataset, the entropy loss encourages the predicted auxiliary label to follow a uniform distribution (thus having a high entropy) when we have a large batch size. Whilst if the prior knowledge of the data distribution is known, we may also further revise the entropy loss to follow a desired distribution by minimising KL divergence as one might wish.

The entropy loss is essential to achieve the human-level performance as shown in further experiments. As it encourages to learn a higher entropy for the auxiliary task, the meta generator will produce a more informative knowledge and effectively avoid local minimas during training in which it produces minimal extra knowledge if directly optimising with raw gradient descent.

3.3 Experimental Results

In Section 3.1, we present the learning behaviour in auxiliary training across different types of network with or without regularisation. Without loss of generalisation, we perform all following experiments in VGG-16 network without regularisation for simplicity.

3.3.1 The Performance of MAXL

We evaluate our proposed MAXL algorithm compared with human-defined hierarchy used in Section 3.1 and single-task training. We exhaustively run all possible hierarchical combinations in our designed 4-level CIFAR100 dataset in Table 3.3 and we present our results in Figure 3.7.

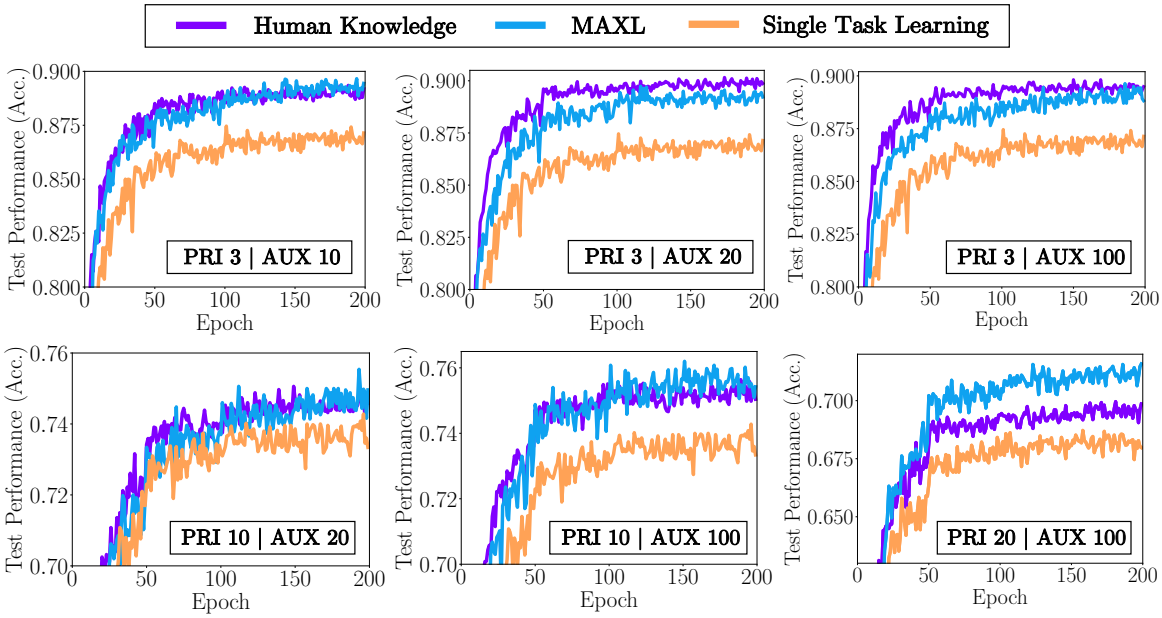


fig. 3.7: The visualisation of learning curves for test data training with MAXL, multi-task learning with human-defined hierarchy and single task learning. We provide all 6 different combinations of primary and auxiliary task in our proposed 4-level CIFAR100 dataset.

Training. We train the multi-task evaluator using the same optimisation method described in Section 3.1.2 with a learning rate of 0.01 and the meta generator with a smaller learning rate 10^{-4}. For both networks, we drop the learning rate by half for every 50 epochs with training for 200 epochs in total and optimise with vanilla stochastic gradient descent. We employ an L_1 norm weight decay of $5 \cdot 10^{-4}$ on meta generator and pick the task weighting of entropy loss to be 0.2 (roughly performs the best during our hyper-parameter grid search).

Results. In Figure 3.7, we observe that our MAXL proposal outperforms the human-defined hierarchy in 4 out of 6 cases. Whilst for those two cases (primary 3-class with auxiliary 20, 100-class) which MAXL do not surpass, it still performs relatively close to the performance with human-defined knowledge and noticeably better than the one with single task training. In addition to training with human knowledge which having a larger performance acceleration in the beginning training steps, our MAXL proposal learns to refine the auxiliary knowledge on the fly and gradually improve generalisation even at the final training stage.

To further explore how knowledge built from MAXL and human can affect generalisation compared to single task learning, we employ t-SNE [MH08] to map our high dimensional features learned in the last shared layer into a 2D map in Figure 3.8.

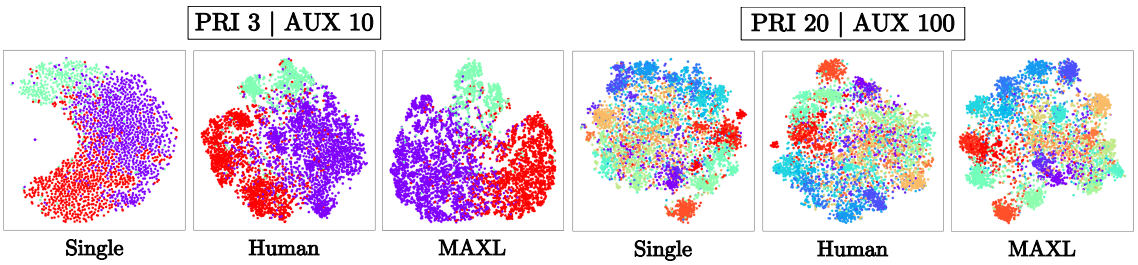


fig. 3.8: t-SNE visualisation of the learned last feature layer in the shared multi-task network training with two primary and auxiliary task combinations from the 4-level CIFAR100 dataset. Colour for each image is defined by the primary class.

There are several interesting findings from the t-SNE visualisation. Within both two cases from Figure 3.8, we can see the MAXL provides the most linearly separable feature representation compared to human-based and single task learning and most of the classes in MAXL are well clustered in a circular shape embedding. To further notice in the case for primary 3-class with auxiliary 10-class, we know both human and MAXL learning produce a similar learning curve as shown in Figure 3.7, whilst the embedding learned in MAXL looks more visually separable.

3.3.2 Explore The Limit of Auxiliary Task Complexity

To understand how the auxiliary class complexity and the entropy loss can affect generalisation, we further design 6 hierarchies with an increasing complexity defined as $\psi[i] = 2, 5, 10, 20, 50, 100, \forall i$ respectively. We perform experiments both with and without entropy loss on CIFAR10 dataset which has primary prediction space of 10.

We provide our results in Figure 3.9 in which we plot the test performance improvement compared with single task learning under different auxiliary class complexities. The performance is provided by averaging the test results from the last 5 epochs to reduce training uncertainties. We also man-

age to calculate class collapsing rate in each hierarchy after training which is defined as the size of collapsed auxiliary predicted space divided by the size of designed predicted space.

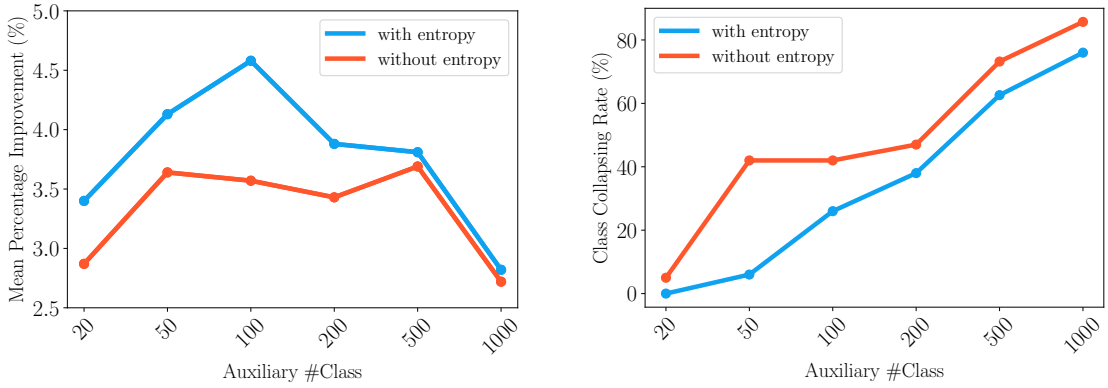


fig. 3.9: Left: Performance improvement in percentages from training with MAXL compared with single task learning in 6 defined hierarchies. Right: The collapsing class rate in percentages for each defined hierarchy.

In Figure 3.9, we show the effectiveness of entropy loss for which the performance improvement can be further increased when we apply the meta objective with entropy loss. Furthermore, we observe an interesting trend in which improvement curves for both with or without entropy loss follows a $\cap$ shape. We believe this trend substantiates the hypotheses we discussed in explaining generalisation in Section 3.1.2. As if we employ an excessive complexity of auxiliary task, the role of auxiliary task simply becomes a pure regulariser by introducing noise during multi-task training. Besides, the experiment shows the auxiliary learning performs the best when we have 100 auxiliary class which is 10 times larger than the primary class and have a relatively small class collapsing rate. This result is also consistent with the one with CIFAR100 in which we show the auxiliary class-20 helps most on class-3 comparing class-10 and 100. In either case, when we have excessive large auxiliary prediction space, it will be collapsed substantially and learns no extra helpful knowledge.

3.3.3 Human Interpretation of Generated Knowledge

To explore the auxiliary knowledge generated by MAXL, we visualise some top predicted candidates in some randomly selected primary classes. We again choose CIFAR100 as an example and

we pick the task combination with primary 20-class and auxiliary 100-class which shows to exceed the performance based on human knowledge most out of other task combinations in Figure 3.7. In addition, we also apply MAXL on an easier dataset MNIST in which auxiliary task is defined to be class-30. We present our results in Figure 3.10.

fig. 3.10: Visualisation of top 5 candidates of 3 randomly selected auxiliary classes in each primary class as shown in the last column. We present the visualisation in CIFAR100 dataset with primary class-20 and auxiliary 100-class (up) and MNIST dataset (down).

To our surprise, the generated auxiliary label visualised in both dataset shows no obvious human understandable knowledge. In particular, we cannot find any similarity within each generated auxiliary class whether in shape, colour, style, structure or semantic meaning. Further, we found out the generated auxiliary knowledge is not fixed since the top predicted candidates are swapped completely when we re-trained the network from scratch. The explanation for such observation is still uncertain, whilst we speculate that the human-designed knowledge is just one out of infinite number of local optimums in knowledge space for each learning task.

Another explanation for the performance improvement is that any auxiliary label generated by MAXL is a soft label which certainly contains more information than the one-hot label defined by human. The benefit of soft label is well studied in [BHRF18] in which the authors showed the model trained with soft labels produced more accurate predictions and more robust to over-fitting.

In our experiments, we show that human knowledge is sub-optimal and there exists a better way for a learning system itself to automatically explore on solving one learning task with meta learning. It remains several questions unanswered like how to interpret machine-generated knowledge and how to close the generalisation gap between human and machine-design knowledge? The answers to such questions require a deeper understanding and analysis in meta and auxiliary learning. We believe it is one of the key steps to understand generalisation in the regime of deep learning.

There's still a long way to go...

Conclusion & Future Work 4

In this thesis, we have investigated several important research aspects in the field of multi-task learning. We start by examining the easy task domination problem in multi-task training which motivates a better design in adaptive multi-task loss functions. We further provide a unified approach by building a multi-task architecture based on attention which can learn task-shared and task-specific features automatically in an efficient and end-to-end manner. Then, we study the problem of auxiliary learning, particularly the generalisation behaviours under different task complexities. Lastly, we design a learning system which is able to generate auxiliary knowledge as useful as human knowledge by combining the advantages and characteristics from both auxiliary and meta learning. Below, we point out several important future directions that should be further investigated.

Multi-Task Structure & Relatedness. In Chapter 3.1, we study auxiliary learning with different task complexities in the same visual domain. Whilst towards a more complicated real-life application, understanding a set of diverse types of tasks in multi domain is far more wanted since most learning tasks are interdependent. In [ZSS+18], the authors studied the multi-task relationships in a graphical structure by producing a computational task taxonomic map. However, the authors achieved the task relatedness only by applying a progressive learning approach as similar to [RRD+16] which is computational inefficient and biased. A more desired approach should explore the inter-relationships in multi-task learning only in one run-time as well as produce a dynamic rather than static complete computational graph since the task relatedness in different training stage might also be different.

Self Auxiliary Learning. In MAXL algorithm, we show that the meta generator is able to generate knowledge to improve performance of the primary task. A further step is considering the case when

having no ground-truth primary task as well. With only providing minimal task prior knowledge, we may develop learning systems which can assist each other on the fly in a self auxiliary learning manner. One practical setting is to build two meta generators in which one meta generator feeds the generated knowledge to the other meta generator to improve performance for a generated task.

Feature Universality. For a fixed parameter space, training with an increasing diverse set of tasks will also increase memory complexity. In [Kok17], authors show that the multi-task network training with 7 tasks drops performance significantly compared with training with 1 task. Having feature representation effectively scales well to a larger number of learning tasks arguably possess a certain level of universality. Whilst design such adaptive learning system is challenging, understanding why and how feature representation scales and behaves to different type of tasks will help us automatically understand generalisation.

The pursuits to understand and create intelligence is a long winding road. It's a road no one knows where it heads or when it ends, but we set foot on this exciting journey anyway. Because we must know and we will know!

Bibliography

[ADG+16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[Azu97] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[BHRF18] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[Car98] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[CBLR17] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask net-

works. *arXiv preprint arXiv:1711.02257*, 2017.

[CFNL13] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.

[CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[CGGS13] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer, 2013.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[DRMS07] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[DZ17] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[EBC+10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[EF15] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE*

International Conference on Computer Vision, pages 2650–2658, 2015.

[EKN+17] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[FNPS16] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[Gav99] Dariu M Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

[GBCB16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[GEB16] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[GHGM14] Georgia Gkioxari, Bharath Hariharan, Ross Girshick, and Jitendra Malik. R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[Hir08] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[IL66] Alekseĭ Grigor'evich Ivakhnenko and Valentin Grigorévich Lapa. Cybernetic predicting devices. Technical report, PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING, 1966.

[JMC+16] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[Kel15] John E Kelly. Computing, cognition and the future of knowing. *Whitepaper, IBM Reseach*, page 2, 2015.

[KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Kok17] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[LGRN09] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[LJD18] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. *arXiv preprint arXiv:1803.10704*, 2018.

[LK18] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018.

[LZCL17] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[MG01] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[MHLD16] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[MPCB14] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[MSGH16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[NSF12] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[RBV17] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *arXiv preprint arXiv:1705.08045*, 2017.

[RD02] Edmund T Rolls and Gustavo Deco. *Computational neuroscience of vision.* Oxford university press, 2002.

[RL16] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[RPK+16] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

[RRD+16] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[Rud17] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Sim95] Herbert A Simon. Artificial intelligence: an empirical science. *Artificial Intelligence*, 77(1):95–127, 1995.

[SRB+17] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for rela-

tional reasoning. In *Advances in neural information processing systems*, pages 4974–4983, 2017.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TBC+17] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4499–4509, 2017.

[TTLL17] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *arXiv preprint arXiv:1704.01631*, 2017.

[Tur09] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[VKP10] DWF Van Krevelen and Ronald Poelman. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9(2):1, 2010.

[Wei66] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[WJQ+17] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.

[WMH+17] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned

optimizers that scale and generalize. *arXiv preprint arXiv:1703.04813*, 2017.

[WTAR10] Christian Widmer, Nora C Toussaint, Yasemin Altun, and Gunnar Rätsch. Inferring latent task structure for multitask learning by multiple kernel learning. *BMC bioinformatics*, 11(8):S5, 2010.

[WZY+12] Jing Wan, Zhilin Zhang, Jingwen Yan, Taiyong Li, Bhaskar D Rao, Shiaofen Fang, Sungeun Kim, Shannon L Risacher, Andrew J Saykin, and Li Shen. Sparse bayesian multi-task learning for predicting cognitive outcomes from neuroimaging measures in alzheimer's disease. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 940–947. IEEE, 2012.

[ZBSL17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6612–6619. IEEE, 2017.

[ZLZ+16] Wenlu Zhang, Rongjian Li, Tao Zeng, Qian Sun, Sudhir Kumar, Jieping Ye, and Shuiwang Ji. Deep model based transfer and multi-task learning for biological image analysis. *IEEE transactions on Big Data*, 2016.

[ZP+14] Bernhard Zeisl, Marc Pollefeys, et al. Discriminatively trained dense surface normal estimation. In *European conference on computer vision*, pages 468–484. Springer, 2014.

[ZSS+18] Amir R Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[ZY17] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

Ethics Checklist

A

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓

	Yes	No
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Legal and Ethical Considerations

B

All the datasets used in this thesis are either internet open license images collected and labeled by human or real-life scenes recorded by commercial sensors. All datasets are for non-commercial use and have been cited with the corresponding references. This research involves neither human and animal based data nor personal privacy data. Since this thesis inclines to a theoretical learning based research and thus involves no commercial applications or the potential of misuse. Further, this thesis is a personal work with no additional collaborators and thus does not involve any developing countries or human participants. Lastly, for further information in data collection in each dataset (e.g. the search and label algorithm, sources and locations of images), please directly contact the original corresponding authors.