Less Labelled Learning
MASTER’S THESIS ARTIFICIAL INTELLIGENCE

LAURENS HAGENDOORN

Master thesis project carried out at TNO

Supervisors at TNO:
MAAIKE DE BOER
HENRI BOUMA
STEPHAN RAALJMakers

Supervisor at Radboud University:
JASON FARQUHAR

Radboud Universiteit
Nijmegen
August 2017
Less Labelled Learning

Abstract

For most of us humans, extracting information from visual data comes very natural. For a computer however, recognising what it is looking at is a challenge. To teach the computer to recognise the objects in an image, it needs to be trained on a large dataset with many annotated images. Creating these annotations is a time consuming and costly process, and the number of unlabelled images available will always greatly outnumber the number of labelled images. We propose a novel method for semi-supervised learning: Ordered ACOL-PL. We show that our method is able to achieve a competitive classification accuracy based on few samples in the training set accompanied by a class label. We also explore the dependencies of the method on the selected and learned feature space, as well as the dependency on the superclasses used.
# Contents

1 Introduction .......................................................... 1

2 Related work ......................................................... 4
   2.1 Object detection ................................................. 4
   2.2 Image classification ........................................... 6
   2.3 Incremental learning ........................................... 7
   2.4 Using the hierarchy ........................................... 10
   2.5 From low- to high-level concepts ............................ 11
   2.6 Low-level concepts ............................................ 13
   2.7 Semi-supervised learning ..................................... 14
   2.8 Deep clustering ............................................... 16

3 Methods ............................................................... 18

4 Experiments and Results ........................................... 23
   4.1 Datasets ......................................................... 24
   4.2 Superclasses and Subclasses .................................. 25
   4.3 Metrics .......................................................... 26
   4.4 Baselines ....................................................... 28
   4.5 Experiment 1 - Training on partially labelled data ....... 29
   4.6 Experiment 2 - Further exploration of relation between ACOL and Ordered ACOL-PL using uninformative superclasses ............ 36
   4.7 Experiment 3 - Varying number of subclasses .............. 38
   4.8 Experiment 4 - Superclass difficulty ........................ 40
   4.9 Experiment 5 - Superclass definitions ...................... 42
   4.10 Experiment 6 - Altering the clustering space - ACOL ...... 46
   4.11 Experiment 7 - Altering the cluster space - Ordered ACOL-PL ... 48

5 Discussion and Future Work ........................................ 51

6 Conclusions .......................................................... 58

References ............................................................. 67
Imagine a parent taking their child to a petting zoo for the first time. The child is fascinated by all the different animals it sees, and is determined to learn all their names. The first five times the child points to a horse and asks what it is, the parent will likely patiently tell their child that the animal it is pointing at is a horse. The next five times the child asks, the parent might start to get mildly irritated, and after the 100th time the child goes: “Hey! Is that a horse too?” the parent is very likely to give up and head home.

This is what training an artificial neural network is like. And where one would expect a child to learn quickly from only a few examples of an object, neural networks need to be provided with thousands of examples in order to achieve a good classification accuracy. Image classification datasets commonly used for deep-learning, such as CIFAR\textsuperscript{20}, ImageNet\textsuperscript{43}, Pascal VOC\textsuperscript{7} or MS-COCO\textsuperscript{28} consist of thousands of meticulously annotated images. One example of this last dataset is shown in Figure 1.1. Clearly, annotating such a dataset takes a large amount of time and effort, and especially when it comes to more specialist datasets, for example a dataset of medical images, this does not come cheap.
Broadly speaking, machine learning comes in three flavours: supervised, unsupervised and semi-supervised. Supervised learning is what requires these large volumes of labelled data, and can be trained to learn the information you want it to learn. Examples of supervised learning are Support Vector Machines or Neural Networks. A very common form of unsupervised learning is clustering. This is a technique by which the inherent structure in the data is used to learn groupings of similar datapoints without having to label a lot of images. A big downside however, is that it is hard to guide the learning process to learn the specific classes you are trying to classify. The final category in machine learning is called partially supervised, or semi-supervised learning. Here, a part of the provided data is annotated, and the rest is not. It thus provides a good blend of supervised and unsupervised learning.

A recent addition to the set of unsupervised learning methods is the Auto Clustering Output Layer (ACOL) proposed by Kilinc and Uysal\textsuperscript{16}. It is an output layer to be placed at the end of a neural network, and is able to learn a clustering based on the features learned by the neural network.

A neural network ending on an ACOL layer can be trained with merely a superclass label, a label denoting in which broad, overarching class the datapoint belongs. No labels of the actual, finer grained, classes that are to be classified are needed. When these superclass labels contain information about the content of the image (informative superclasses), one could argue ACOL is semi-supervised. However, ACOL could also be used with an uninformative superclass, where for example the superclass only indicates whether a certain trans-
formation was applied to the input image or not, in which case the learning process is fully unsupervised. A downside of ACOL is that it merely generates a clustering that optimises the specified loss function, and these clusters do not necessarily correspond with the desired classes. The network might for instance learn that separating the input based on the colour of the third pixel makes for a great clustering. If the goal is to place a variety of animals each in their own cluster, this behaviour is far from desired.

In this thesis we propose a novel method for semi-supervised learning based on ACOL: Ordered ACOL-Partially Labelled (Ordered ACOL-PL). To guide the clustering process, a small subset of the data is annotated. First the network is trained with only this small percentage of labelled data. These labelled instances are quickly learned by the network, and classified correctly with high accuracy. However, generalisation becomes a challenge when the network is only trained on a small amount of labelled data. To remedy this, this first training stage acts as a seed during the next training stage where the network is provided with both the labelled and the unlabelled data. This way the network is able to more easily learn what the intended clustering is, and it is aided in learning a feature space where these desired clusters are separable. We show that our method is able to achieve competitive classification accuracy based on few samples in the training set accompanied by a class label, given the right conditions. We also explore the dependencies of the method on the selected and learned feature space, as well as the dependency on the superclasses used.

Chapter 2 provides an overview of related work and the current state of the art. In Chapter 3, our method is described and implementation details are provided. Chapter 4 reports on a variety of experiments. Here, first some of the parameters and datasets used are discussed. It is then shown that Ordered ACOL-PL performs competitively, under the condition that ACOL produces a desired clustering. The other experiments inspect various aspects of ACOL and Ordered ACOL-PL and provide useful insights into how the performance could be improved. In Chapter 5, some of the limitations of this work are discussed and future paths to take are elaborated upon. Finally, Chapter 6 contains the conclusions.
No good work stands on it own, this section will embed Ordered ACOL-PL in the field. First several object detection networks will discussed. This is followed by image classification networks. Subsequently incremental learning techniques are discussed, Ordered ACOL-PL could be used towards this end, as will also be discussed in the discussion (Section 5). This is then followed by some methods that could be used to utilise the hierarchy that Ordered ACOL-PL automatically discovers. This hierarchy is used to go from low- to high-level concepts, or from fine-grained to coarse labels, the state of the art of this step is discussed, as well as several possible representations of the featurespace in which this should be done. Then an overview will be given of the current state-of-the-art of semi-supervised learning, to which Ordered ACOL-PL is a new addition. Finally, several neural network based clustering methods are discussed, one of which forms the basis of Ordered ACOL-PL.

2.1 Object detection

The task of object detection in images is a well known task within machine learning and the state of the art is evolving quickly. This state of the art pri-
arily consists of neural networks. These networks come in a large variety of shapes and sizes, each variant with its own benefit over others. Object detection requires not only recognising the objects in an image, but also localising where in the image these objects are. Usually the output of an object detection network consists of a set of bounding boxes, each accompanied by a classification. A harder variant on these bounding boxes is image segmentation, where the objects are given a pixel-level mask. Various challenges focus on the field of object detection, and at these challenges the strongest networks and techniques for object detection can be found. In the ImageNet\textsuperscript{43} competition of 2016, the object detection category was won by Zeng et al.\textsuperscript{57}. In their paper, they propose a Gated Bi-Directional Convolutional Neural Network (GBD-Net), which is a variation upon a Fast Region-based Convolutional Neural Network (Fast-RCNN)\textsuperscript{10}, where different layers pass messages to each other. This is done in order to utilise the relations between positional and visual cues for feature learning. The second place of the ImageNet competition that year went to a Faster-RCNN\textsuperscript{42} implementation. The 2016 MS COCO\textsuperscript{28} detection challenge was won with an ensemble of five Faster-RCNNs. A Faster-RCNN consists of a Region Proposal Network combined on top of any image classification network. The features found by the image classification network are used by the RPN to find a large number of possible bounding boxes, after which a small network classifies the object in each of these bounding boxes.

A few different architectures are found in the Pascal VOC challenge\textsuperscript{7}. The best performance on this dataset is achieved by a Single Shot multibox Detector (SSD)\textsuperscript{29}, followed by a Faster-RCNN implementation, and then followed by YOLOv2\textsuperscript{41}. YOLOv2 and SSD are a end-to-end fully convolutional systems for object detection, which learns both the bounding boxes and the classification directly. Faster-RCNN differs in this aspect, as it uses a separate network for proposing Regions Of Interest (ROI), that shares convolutional layers with the classification network. The PASCAL VOC has a separate category where the use of additional data, not provided by the challenge, is permitted. Here, a Region-based Fully Convolutional Neural Network (R-FCN)\textsuperscript{25}, a fully convolutional variant on Faster-RCNN, performs far better than YOLOv2 (88.4 MAP
versus 78.2). In fact, in this category the best six submissions are Faster-RCNN based.

For object detection in images, without additional segmentation requirements, Faster-RCNN still seems to be able to deliver state of the art performance. Other architectures, albeit more recent, do not necessarily provide improvements on the image detection performance. R-FCN mostly improves over Faster-RCNN in its ability to generate segmentations. Faster-RCNN performs especially well when used in an ensemble of networks. When comparing single networks YOLOv2 and SSD outperform Faster-RCNN in both accuracy and speed. Faster-RCNN provides a more flexible architecture with clear separation between features and bounding boxes. All these techniques are able to quickly classify the objects in an image, and locate them using a bounding box.

Ordered ACOL-PL could be a useful addition to these object detection methods, because it can detect different classes within each class in the dataset. It could for example automatically cluster a given class ‘car’ into different kinds of cars, or cars with different orientations, each of which will have significantly different bounding boxes that can then be determined specifically for the subclass. These more narrow classes are likely also easier to classify, increasing the overall classification accuracy of the superclass. While implementing this is left as future work in this thesis, Ordered ACOL-PL in its current state does allow for performing object detection using fewer annotated examples.

2.2 Image classification

A subset of the object detection task is the image classification task. Hereby the entire image needs to be assigned the labels of all the objects in the image. These objects do not need to be localised, and generally each image only contains a single object in the centre of the image. Aside from ImageNet, other examples of these types of datasets are CIFAR and MNIST. On these image classification datasets a different set of networks is used, as the detection capabilities are no longer needed. Historically VGG16 by Simonyan and Zisserman generally performs very well, and is still commonly used as the basis for,
for example, Faster-RCNN. VGG16 uses small, 3 by 3 convolutions to create deep neural networks. Szegedy et al.\textsuperscript{47} proposed Inception networks, which are a more recent example of successful image classification networks. Inception networks go even deeper than the VGG16 architecture by employing layers that consist of small networks themselves. Xception by Chollet\textsuperscript{4} is an iteration upon Inception networks that uses depthwise separable convolutions instead of the inception modules. Another branch improving deep learning can be found in RESNETs by He et al.\textsuperscript{14}, that use narrow architectures working on the residuals of a mapping to efficiently create and train very deep networks. Since VGG16 is still a common benchmark network, and provides an easy to adapt architecture, Ordered ACOL-PL will be placed on the VGG16 network for work in this thesis. Note that in theory Ordered ACOL-PL can be easily attached to any architecture.

While these methods show very impressive performance, they have one downside: the architectures as-is are not able to incrementally learn new classes. In order to introduce it to a new class, the network will need to be adapted, and portions of the network will need to be re-trained.

2.3 Incremental learning

Due to the fact that real world data constantly evolves and changes, incremental learning is a desired capability. Incremental learning enables a network to keep up with continuous changes in currently known concepts, as well as to add entirely new ones. Incremental learning comes in roughly two flavours. In the first an intermediate feature space is learned, and from there new classes are learned as combinations of these. In the second, a neural network’s architecture is modified to accommodate the new class.

2.3.1 Using an intermediate feature space

One example of a system using the first method, is iCaRL\textsuperscript{40}. In their paper, Rebuffi, Kolesnikov and Lampert propose a system that learns a feature representation using a CNN and classifies using a method they call Nearest-Mean-
of-Examplars Classification (NMC). They build a list of inputs for each class that is as representative as possible of the mean vector of the class. By storing these inputs rather than storing the mean, the mean vector is automatically updated as the CNN learns and adapts to the new data. This technique is not nicely end-to-end but consists of several stages.

Another method that makes use of an intermediate representation is proposed by Gupta and Chaudhury. They describe a system where they extract features using a CNN. These features are then classified into attributes of the eventual classes they wish to detect, as defined by a pre-defined ontology. The label that is then assigned is that of the class that has the highest average score of its leaf nodes. The classification from features to attributes is done through the use of another (single-layered) neural network, Gupta and Chaudhury define an objective function for this network that takes the ontology into account. Although they propose the method for use in transfer learning, due to the used ontology, it could be extended for use in incremental learning.

Fu and Sigal use a technique called semi-supervised vocabulary-informed learning. It works by minimising the distance between an image and its label in a semantic word space. While the paper does not propose it as an incremental learning method, it could be adapted to work incrementally. One problem however is that, when adding a new class, it is unknown where in the semantic word space its label should be placed.

Bouma et al. propose a class incremental learning technique where they create a hierarchy by joining two classes underneath a single superclass when these two classes heavily draw hard-negatives from each other. This way the proposed method is able to automatically discover a hierarchy based only on visual features. Additionally, new classes can be added on the fly, since each class is classified by their own SVM, based on features extracted by a CNN, and there is minimal interaction between different classes.
2.3.2 Adapting the neural network architecture

Xiao et al.\textsuperscript{49} use the other type of incremental learning, where the architecture of a neural network is adapted to accommodate the new classes. They propose to dynamically grow a neural network, layering it to first classify into a coarse super-class, followed by classifying into finer-grained sub-classes. Although this method seems very natural, with an error rate of 48.52\% on a subset of just the 2200 animal classes in ImageNet, the performance is not quite state of the art. They do show a performance increase from learning incrementally over learning from scratch. This performance increase becomes increasingly smaller however as more new classes are added. This is due to obvious scalability issues, as their method constructs very large networks and has many weights to learn.

Yan et al.\textsuperscript{52} propose a method that shows an improved performance on ImageNet. In their architecture they extract low-level features with a shared network component. From there they make coarse class predictions, which are either used directly or combined with the low-level features and used as input for a range of networks each classifying for a small set of sub-classes. The architecture proposed by them is not directly appropriate for incremental learning, but due to the hierarchical nature this could be done without retraining the network in its entirety. This is similar to the method shown by Agethen and Hsu\textsuperscript{1}. They train “expert networks” for each new (set of) classes added, as well as a mediator network learning to combine these. Extending this framework to incremental learning would only involve adding another expert network.

Growing the network is also proposed by Käding et al.\textsuperscript{15} and by Li and Hoiem\textsuperscript{26}, the latter of which proposes an alternate learning scheme as well. These approaches do not consider a hierarchical structure, but just add new nodes to the final layer. Using this naïve approach would mean that learning subclasses would likely conflict with other subclasses it knows and with the superclass. Subclasses within the same superclass are likely very similar to each other and to their superclass. Adding them as yet another class, without taking this into account is likely to degrade the performance of the classifier. The problem remains that growing a network will lead to increasingly more difficult classifica-
tion involving more weights.

The effect of an increasing number of weights could be mitigated using the work by Kruithof et al.\textsuperscript{22} They showed that given two disjoint datasets, a network trained on one could transfer well to the other. This is done by entirely re-training it in case the new dataset has plenty of training samples. Or, in case there are very few training samples available, by copying the network trained on the first dataset, freezing the first several layers, and re-training only the last few layers. This shows that the first layers in a neural network learn a general representation, largely shared over datasets. As such, many of the weights do not need to be learned from scratch, or not at all if their layers are frozen, and expanding networks is less of a problem in that respect.

The similarities between related classes, and the fact that their labels are not always mutually exclusive, make classification more challenging. Given enough training data, as is the case with a large dataset such as ImageNet, these challenges could be overcome by using just a neural network. With very few examples per class however, the use of a hierarchy will be beneficial, as it would allow the network to use knowledge about the neighbours as well. Even though growing a neural network does introduce some scalability issues, the fact that the system can be optimised using backpropagation and is contained in a single system is a benefit.

Ordered ACOL-PL could prove to be very useful in the domain of incremental learning. A hierarchy is essentially a cluster that is in turn divided into new clusters. Using Ordered ACOL-PL, these hierarchies could be discovered automatically, and the clustering is able to efficiently classify with only a limited number of examples for a class.

2.4 Using the hierarchy

A discovered hierarchy, in addition to facilitating the incremental learning with classes that are not mutually exclusive, also allows for boosting the classifier performance. YOLO9000\textsuperscript{41} is an extention of YOLOv2 that uses a hierarchy to merge the knowledge that can be learned from vast classification task datasets.
with object detection datasets. That way, the network is able to correctly de-
tect a large number of classes it has not previously seen bounding boxes for.
Goo et al.\cite{11} propose a difference-pooling operation to emphasise the difference
between parent and child classes, leading to increases in classification perfor-
mances. Marino, Salakhutdinov and Gupta\cite{34} propose a Graph Search Neural
Network, that incorporates a knowledge graph in an end-to-end system. And
Srivastava and Slakhutdinov\cite{46} use a supplied hierarchy for setting priors on re-
lated classes; this boosts performance when one of these classes has very few
training examples. Furthermore, having this hierarchy could allow for easier in-
terpretation by the user.

Ordered ACOL-PL automatically discovers a small hierarchy when it is clus-
tering. As it takes one overarching superclass and divides this into several child-
classes, or subclasses. If desired, this could be repeated, clustering the subclasses
into subclasses of their own, forming a hierarchy that way. Ordered ACOL-PL
uses a loss function to spread out the subclasses within one superclass, but in-
stead of doing this based on the content of the image, it uses the activations of
the clustering nodes.

2.5 From low- to high-level concepts

ICaRL and some of the other incremental learning techniques rely on some in-
termediate step, going from low-level concepts to higher-level objects, or from
coarse classes to more fine-grained objects. This step is also performed by Or-
dered ACOL-PL, where the features are used to classify into low level objects,
which are then in turn used to classify into overarching superclasses. The state
of the art of making this step from low-level to high-level concepts can be found
in settings such as TRECVID\cite{2}, in this challenge on Multi-media Event Detec-
tion (MED) participants are tasked with detecting high-level events from videos.

The VIREO team\cite{58} in the TRECVID MED challenge of 2016 uses two tech-
niques to go from concepts to events. In the first technique they construct a
large concept bank. These concepts are then used for event classification using
a Chi-Square SVM. The second method VIREO uses uses VLAD-encoding\cite{51} to
pre-process the frame as well as the individual ROI’s, and classifies the events from those. They show the second method works better than the first, but an ensemble of both performs even better.

The NTTFudan team at TRECVID MED do not create any sort of intermediate representation but simply extract as many features as possible, and apply an SVM for each individual event. This works very well, although it requires a larger amount of training data.

The Mediamill team performed very well at the TRECVID MED challenge by using two different embeddings. One embedding maps to a representation made using VideoStory, a method through which both the video and the description is used to more accurately represent the events in the video. The other embedding uses an ImageNet Shuffle. This means that they used a network that was pre-trained to the 22,000 ImageNet classes. As many of these classes are far too specific to be useful in this task, they merged some of the concepts using WordNet.

Finally, Team INF from Carnegie Mellon University also extracts a very large set of features from the input, and propose a self-paced curriculum training scheme. This works fairly well given enough samples, but performance in a fewer-example scenario is not as good. In all these cases it is clear that going from low-level concepts to high-level events is relatively straightforward given a strong low-level representation.

The aforementioned methods for crossing the gap between low- and high-level concepts all do so by describing the concepts in such a way that they can be classified into the events. Other techniques, such as presented by Markatopoulou et al., consist of extracting possible concepts from the event query, and computing the distance from these to each of the concepts in their concept bank. They then select a number of defining concepts. All videos are tagged with the concepts they contain as identified by a classifier, and with these tags they can then be retrieved for the specific events. Although doing it this way also gave satisfactory results, in many scenarios such queries accompanied with elaborate descriptions as provided in the TRECVID MED challenge are not available.
2.6 Low-level concepts

As established in the previous section, transitions from low- to high-level concepts can be relatively trivial, given a strong representation of the low-level concepts. In order for Ordered ACOL-PL to perform a correct clustering, it needs a strong representation of the low-level concepts. Because this feature space is guiding the type of clusters formed by Ordered ACOL-PL, it is important to understand the type of features learned by a neural network. There are several ways to create strong low-level concepts and to gain insight in them.

One possibility, as shown by the majority of the TRECVID MED participants, is to use a large concept bank. Such a concept bank would consist of the output of the final layer of a CNN trained on a large set like ImageNet. A larger concept bank could be created by combining the outputs from a network trained on different datasets, with different concepts to be classified. The concept bank could also be extended using the features from further down the neural network.

Another interesting possibility would be the use of patches as visualised by Zeiler and Fergus\textsuperscript{56}, CNNs learn different kernels that react strongly to very specific patterns, such as a dog’s nose, a horse’s leg, or a car door. If a low-level representation would include patches that respond strongly to wheels, and others that have a strong response to car windows, these responses intuitively correspond to a car. More specific cars further down the hierarchy would then consist of many of the same responses to the patches, but a type of car that always has a large grill on the front could be separable by a patch that responds strongly to such a pattern. Mahendran and Vedaldi\textsuperscript{33} visualise the inner workings of CNNs in three different ways. In inversion the network is used in the opposite direction, using the output as input. For activation maximisation, an input is found that maximally activates a neuron. The final way is through caricaturisation, where the network is asked to maximise the different activations it has given a certain input.

Yosinski et al.\textsuperscript{54} show two tools they created to visualise CNN. From the visualisations they present, it becomes clear that, as intuition suggests, the acti-
vations from the convolutional layers are quite general. The activations of the fully connected layers become more and more localised however, and they start to combine the different patches that the convolutional layers recognise.

Oquab et al.\textsuperscript{37} successfully use the final fully connected layer (fc7) of AlexNet\textsuperscript{21} trained on ImageNet, as a feature representation for classification with a shallow, two-layered, neural network on Pascal VOC. This shows that previously learned patches, that in this case are found in the second fully connected layer after the convolutions, can be used for this purpose.

Finally Yu et al.\textsuperscript{55} visualise the activations in different convolutional layers. They show that deeper networks are better in removing the background and focus on the essence of what defines a class in their lower convolutional layers. On the one hand this might mean the output of these layers might be well suited as a low-level feature, on the other hand it might hinder the detection of the larger whole that these patches are a part of.

It appears that the features a neural network creates generally already work quite well as a strong low-level representation, and Ordered ACOL-PL will not use any additional methods such as the creation of a large concept bank created from the concepts found by a combination of networks.

2.7 SEMI-SUPERVISED LEARNING

Classification networks are usually trained with a vast amount of data, and all of it is usually accompanied with a class label. This is called supervised learning. As annotating all these images is generally very labour intensive, desirable alternatives are unsupervised or semi-supervised learning, whereby all or part of the data is not accompanied with a class label. These types of learning problems are currently mostly handled in three different ways; using Self-labelled learning, using generative models or using graph-based methods. Ordered ACOL-PL is a semi-supervised method, related to graph based methods, only requiring a small portion of the dataset to be labelled to still achieve a competitive classification accuracy.

One way of doing semi-supervised learning, is to train a model with the avail-
able labelled data, and assign part of the unlabelled data a label based on the classification of this network. This can then be repeated until all the data is confidently labelled. Methods of this type of semi-supervised learning are appropriately called Self-labelled techniques. Triguero, García and Herrera\textsuperscript{48} provide an extensive review of many of such methods. As one of the top methods they identify TriTraining by Zhou and Li\textsuperscript{60}. This method trains three classifiers on the labelled data, whenever two classifiers agree on a label for an example, it is labelled for the third. Another method Triguero et al. name as one of the better methods is Democratic co-learning by Zhou and Goldman\textsuperscript{59}. This is a method that also uses multiple classifiers to perform majority voting on the unlabelled part of the dataset.

Another way of doing semi-supervised learning is using generative models. Kingma et al.\textsuperscript{19} and Maaløe et al.\textsuperscript{31} propose the use of deep generative models to perform semi-supervised learning, they learn a probabilistic feature space using a neural network. In this new feature space different classes are more easily separated by an SVM or a clustering algorithm.

Papandreaou et al.\textsuperscript{38} use a combination of fine-grained labels and coarse (weak) labels for semi-supervised learning, this is similar to Ordered ACOL-PL. Papandreaou et al., however, use it for semantic image segmentation and use an expectation maximisation method of combining the labels. Ordered ACOL-PL is a method used for image classification, and the coarse labels are used as a clear objective for the neural network.

Rasmus et al.\textsuperscript{39} proposed a network that minimises a special loss function to enable semi-supervised learning. In this case this loss function reduces the noise within each class, in addition to performing supervised classification for the labelled data. Although both this method and Ordered ACOL-PL mainly contribute a loss function to enable the semi-supervised learning, the type of loss function is quite different. Another important difference is that Rasmus et al. use an auto-encoder architecture, whereas Ordered ACOL-PL does not need this additional complexity.

A set of methods closely related to self-labelling techniques are graph-based methods. These techniques form a graph where each labelled or unlabelled data-
point forms a node. A regulariser is then applied to the graph to create a structure through which unlabelled datapoints can be assigned their labels. A system of this kind is proposed by Fergus, Weiss and Torralba\textsuperscript{8}, who propose to use anchor points to more efficiently handle these types of algorithms. This idea is further reviewed by Liu, Wang and Chang\textsuperscript{30}. Another semi-supervised technique that uses a graph based method is GAR\textsuperscript{17} by Kilinc and Uysal. It uses a loss function similar to the loss used in Ordered ACOL-PL, as well as a two-step training scheme, where it is first trained using only the labelled data and later only with the unlabelled data. GAR does not, however, utilise a so called ‘coactivity’ term, which Ordered ACOL-PL does use. GAR also does not use superclasses for guidance during the second training phase, but rather it relies on its pre-training to guide this second phase. Furthermore, their activity regularisation, here the ACOL clustering loss, is not used during the first stage of training, whereas this loss is applied during the first stage of Ordered ACOL-PL.

2.8 **Deep clustering**

Most commonly used clustering algorithms have been formulated a long time ago. Since then, deep learning has quickly risen to be the dominant way of classifying images, as evident by their dominance in image classification challenges. Clustering methods are often applied to a feature space extracted by a trained deep neural network, as these generally provide efficient embeddings of the input. The problem with doing this completely unsupervised is that in order to train a network, a clear (and meaningful) objective is required. In a supervised setting, this objective is a class label, but these are not present in an unsupervised scenario such as clustering. To solve this problem one commonly used method is the use of auto-encoders. These auto-encoders can create an efficient embedding of an image, with the image itself as the objective. One method based on such an auto-encoder is Deep Embedded Regularized Clustering (DEPICT) by Dizaji et al.\textsuperscript{6}. DEPICT uses an auto-encoder to learn a representation of the unlabelled images. A softmax layer is then placed on this
learned representation, and through the use of a balanced cross-entropy layer a clustering is learned. Another deep learning based clustering method using an auto-encoder architecture is Deep Embedded Clustering (DEC) by Xie, Girshick and Farhadi\textsuperscript{50}. They use an auto-encoder to initialise the network, and follow this with a self-labelled technique assigning to learned clusters. This same structure is also followed by Yang et al.\textsuperscript{53}, but their proposed method relies on self-labelling via k-means and requires complex learning schemes, such as layer-wise pre-training in order to produce desired clusterings.

An alternative to an auto-encoder for learning to cluster with a deep neural network is to use a special loss function. One example of this is Deep Spectral Clustering Learning (DSCL), proposed by Law, Urtasun and Zemel\textsuperscript{23}. DSCL learns a feature representation whereby the distance of similar objects is minimised. This is done by iteratively optimising the feature space and applying k-means to find the similar objects. Another method that does not use an auto-encoder but instead an adapted loss function is the Auto Clustering Output Layer (ACOL) by Kilinc and Uysal\textsuperscript{16}. In contrast to DSCL this method does not use an alternating scheme of clustering and learning the feature representation, but this is all embedded into a single network. As ACOL provides the most flexibility out of all these clustering methods it has been chosen as the basis of Ordered ACOL-PL.
If one sheep went over the dam, more will follow.

A common Dutch saying

3

Methods

Our proposed method, Ordered ACOL-Partially Labelled (Ordered ACOL-PL), is an adaptation of the Auto Clustering Output Layer (ACOL) as proposed by Kilinc and Uysal\textsuperscript{16}. ACOL is a method to determine a clustering using a deep neural network. It is an output layer that can replace the softmax layer of any regular architecture. When the method is provided with input only labelled with a superclass label the ACOL layer learns to cluster this input by minimising a loss function. The ACOL loss consists of a combination of several regularisers which together enforce a clustering. The three most important terms in this loss function are balance, which ensures that the input is spread out over the clusters, and coactivity and affinity, which ensure that the clusters are specialised.

Balance ($\beta$) is a term that ensures that all nodes in the clustering layer are activated roughly the same number of times within one batch. Without this term all the input samples would simply pass over one of the nodes to be classified into their superclass. The network has no intrinsic use for the other redundant nodes, as the superclass classification is simply a pooling over the clusters. For example, take a $n \times k$ matrix to represent the cluster layer, where $n$ is the number of superclasses and $k$ is the number of clusters. The prediction for the
superclass is then obtained by taking the maximum value in each row, i.e. the maximum value over all clusters per superclass. The information about which column (cluster) has the highest value is discarded by this operation, and the inputs are only classified into one of the rows (superclasses), using a single node. The balance term counteracts this by rewarding the network when it uses more clusters. Equation 3.2 shows the function for balance, where $k$ is the number of nodes in the clustering layer (i.e. the number of superclasses times the number of clusters) and $V$ is a $k \times k$ matrix as shown in Equation 3.1. Here, $Z_{s,a}$ is the activation of node $a$ for sample $s$. $V$ then denotes how often any combination of two nodes fires within one training batch.

$$V = \begin{pmatrix} 
\sum^k Z_{i,1} & \sum^k Z_{i,1} & \sum^k Z_{i,1} & \cdots & \sum^k Z_{i,1} \\
\sum^k Z_{i,2} & \sum^k Z_{i,2} & \sum^k Z_{i,2} & \cdots & \sum^k Z_{i,2} \\
\vdots & \vdots & \ddots & \cdots & \vdots \\
\sum^k Z_{i,k} & \sum^k Z_{i,k} & \sum^k Z_{i,k} & \cdots & \sum^k Z_{i,k} 
\end{pmatrix}$$  

(3.1)

$$\beta = 1 - \frac{\sum_{i \neq j} V_{ij}}{(k - 1) \sum_{i=j} V_{ij}}$$  

(3.2)

Coactivity ($\alpha_c$) is a term that rewards specialisation of the nodes. With only the balance added to the loss, the network will more or less randomly distribute the input over the nodes within each row. By minimising coactivity, the network is penalised for having multiple nodes with a high activation for a single input. The formula for coactivity is show in Equation 3.4. $U$ is a $k \times k$ matrix as shown in Equation 3.3.

$$U = \begin{pmatrix} 
\sum^k Z_{i,1}Z_{i,1} & \sum^k Z_{i,1}Z_{i,2} & \cdots & \sum^k Z_{i,1}Z_{i,k} \\
\sum^k Z_{i,2}Z_{i,1} & \sum^k Z_{i,2}Z_{i,2} & \cdots & \sum^k Z_{i,2}Z_{i,k} \\
\vdots & \vdots & \ddots & \vdots \\
\sum^k Z_{i,k}Z_{i,1} & \sum^k Z_{i,k}Z_{i,2} & \cdots & \sum^k Z_{i,k}Z_{i,k} 
\end{pmatrix}$$  

(3.3)

$$\alpha_c = \sum_{i \neq j} U_{ij}$$  

(3.4)
Affinity ($\alpha$) is the normalised version of the coactivity term, this is added to work better in combination with the normalised balance term. The affinity formula is shown in Equation 3.5. At first the affinity is minimised and the coactivity is unused. The coactivity is only added to the loss function when the affinity dips below a threshold (in this thesis a threshold of 0.03 will be used, as Kilinc and Uysal\textsuperscript{16} recommend). At that point the coactivity has a value that is small enough to not entirely disrupt the training. These three terms: balance, coactivity and affinity, respectively ensure spread and specialisation, which results into a clustering of the input.

$$\alpha = \frac{\sum_{i \neq j} U_{ij}}{(k - 1) \sum_{i=j} U_{ij}}$$

As stated, our proposed method builds upon ACOL. Our adaptation lies in part in the way the network is trained, and in part in a adaptation of the loss of ACOL. The training is first done on just the small set of labelled training data, the network learns a basic idea of the classes it is supposed to classify during this stage. It is then followed by the second training stage, where all training data, labelled and unlabelled, is used. This second stage allows the strong classification seed on the limited train set from the first stage to be generalised better to unseen data, as it is trained with a larger variety of images. The unlabelled data in this stage is clustered around the basic concepts of classes already formed during stage 1. The loss function is adapted by adding a cross-entropy over the clustering layer in the ACOL architecture, allowing the network to use the subclass labels as well as the superclass labels, transforming ACOL into a semi-supervised learning method. The labels for this cross-entropy (the subclass labels) are provided as a one-hot-encoded 2D-matrix corresponding to the 2D clustering layer. The loss function used by ACOL is shown in Equation 3.6.

$$\text{loss} = c_0 \gamma_{\text{superclass}} + c_1 \alpha + c_2 (1 - \beta) + c_3 \alpha_c + c_4 \|Z\|_F^2$$

Most of the notation follows Kilinc and Uysal\textsuperscript{16}: $c$ are weight coefficients, $\alpha$ is the affinity, $\beta$ the balance, $\alpha_c$ is the coactivity, $\|Z\|_F^2$ is the $L_2$ regularisation
term over the activations going into the clustering layer, and finally $\gamma$ denotes the cross-entropy. Our Ordered ACOL-PL method uses the loss shown in Equation 3.7, which differs from Equation 3.6 in the subclass cross-entropy loss.

\[
loss = c_0\gamma_{\text{ superclass}} + c_0\gamma_{\text{ subclass}} + c_1\alpha + c_2(1 - \beta) + c_3\alpha_c + c_4\|Z\|_F^2 \quad (3.7)
\]

Ordered ACOL-PL can be applied on any neural network architecture. For this thesis two architectures using Ordered ACOL-PL are created: one for each dataset used in the experiments. These datasets are MNIST\textsuperscript{24} and CIFAR\textsuperscript{20}. Both these networks are trained using the Adam optimiser\textsuperscript{18}, and the parameters for the ACOL layer loss are equal to those used by Kilinc and Uysal\textsuperscript{16}. One point of attention is that, following Kilinc and Uysal, for superclass definitions where ACOL clusters into more than six clusters, the threshold for the coactivity to be activated is set to 0.06, for all other superclasses a threshold of 0.03 is used.

The architecture used on MNIST is shown in Figure 3.1. This network architecture is based on the tensorflow tutorial Deep MNIST for Experts*, which provides an easily adaptable network that achieves a classification accuracy of 99.2%. Even though it would be possible to achieve a higher performance with different methods, the achieved accuracy is sufficient for the use in this work, as only variations on this network are compared, meaning their relative performance is important. Each convolutional and fully connected layer is followed by a ReLu. Our additions are placed on top of the ReLU after the fully connected layer before the classification layer in the standard network (FC1 in Figure 3.1). The addition consists of the Dropout Layer and the ACOL components (ACOL, stack clusters, matrix softmax and reduced softmax). The MNIST networks are trained in batches of 128 images, with a learning rate of $10^{-5}$.

For classification on CIFAR, Ordered ACOL-PL was placed on top of a VGG16 network that was pretrained on ImageNet. Ordered ACOL-PL was placed after a dropout layer appended to the non-linearity (RELU) on the second-to-last fully connected layer (Fc7) and the convolutional layers are frozen. The Fc6 and

*https://www.tensorflow.org/get_started/MNIST/pros
Figure 3.1: The architecture of the network used for experiments on MNIST.

Fc7 layers are trained on CIFAR, but are initialised with the weights learned on ImageNet. In order to be able to use this initialisation of the fully connected layers, the input of CIFAR was scaled from $32 \times 32$ to $244 \times 244$ pixels, and the ImageNet mean colour was subtracted to match the ImageNet input. This way the input from the CIFAR dataset matches the ImageNet input. To further augment the dataset, about half the images in each batch were randomly flipped horizontally. The network was trained in batches of 48 images with a learning rate of $10^{-6}$.

On both datasets the network was first trained on only the part of the data that has subclass labels for 30,000 batches, followed by training on all data for 20,000 batches. These number of iterations have empirically shown to generally be enough for each stage to converge fully. After the first stage of training, Ordered ACOL-PL has learned features that lead to a classification of the subclasses that does as well as possible on the train set. After the second training stage, the network has seen much more data, and it learns to generalise better to the unseen test data. All code has been provided on github\(^\dagger\).

\(^\dagger\)https://github.com/TheLaurens/Thesisproject/
Experiments and Results

This section elaborates on the results and the setup of several experiments. First it discusses the used datasets, metrics and baselines. It is then shown in Experiment 1 how Ordered ACOL-PL outperforms the used baselines, but also some conditions are encountered under which it does not. In Experiment 2, it is further established that the performance of Ordered ACOL-PL is correlated with the performance of ACOL. In Experiment 3, it is determined that the number of clusters used does not play a major role in any of the performed experiments. Experiment 4 explains the difference in performance of the ACOL methods between the different MNIST superclasses. Then Experiment 5 applies this explanation to CIFAR and provides more insight into the effect of various superclasses. Experiment 6 further explores ways of improving ACOL, which indirectly improves Ordered ACOL-PL. This is done by changing the feature space the clustering is applied on. Finally, in Experiment 7, variations on Ordered ACOL-PL are tested. The hypothesis for each of these experiments are shown in Table 4.1.
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1 (Section 4.5)</td>
<td>We hypothesise that Ordered ACOL-PL will perform similar to direct classification when many labels are available, and that it will outperform all baselines when few labels are available.</td>
</tr>
<tr>
<td>Experiment 2 (Section 4.6)</td>
<td>We hypothesise that the subclass accuracy of Ordered ACOL-PL is dependent on ACOL for few labels on an uninformative superclass definition.</td>
</tr>
<tr>
<td>Experiment 3 (Section 4.7)</td>
<td>We hypothesise that the number of subclasses ACOL needs to classify per superclass does not explain the difference between different superclass definitions with a varying number of subclasses.</td>
</tr>
<tr>
<td>Experiment 4 (Section 4.8)</td>
<td>We hypothesise that the informative superclass definition requires more knowledge about the individual classes than the uninformative superclass definition.</td>
</tr>
<tr>
<td>Experiment 5 (Section 4.9)</td>
<td>We hypothesise that the superclass definition largely determines the type of clusters formed by ACOL.</td>
</tr>
<tr>
<td>Experiment 6 (Section 4.10)</td>
<td>We hypothesise that ACOL is capable of selecting the features relevant for the desired clustering.</td>
</tr>
<tr>
<td>Experiment 7 (Section 4.11)</td>
<td>We hypothesise that Ordered ACOL-PL, despite having additional information available for the feature selection, also is not aided by manual feature selection.</td>
</tr>
</tbody>
</table>

### 4.1 Datasets

#### 4.1.1 MNIST

The experiments will partially be conducted on the MNIST dataset. This well known dataset consists of 60,000 images of handwritten digits between zero and nine. Of these images, 45,000 form the train set, 5,000 images form the validation set, and the final 10,000 images form the standardised test set on which
the results in this thesis are reported. All images are $28 \times 28$ pixels in size and are centred. As this dataset is easy to classify correctly, and consists of very small images, this dataset allows for quick iteration. MNIST is a commonly used dataset and it is therefore useful for showing the benefits of our proposed methods. Some examples of the images in the dataset are shown in Figure 4.1.

![Figure 4.1: Some examples of the images in the MNIST dataset.](image)

### 4.1.2 CIFAR

Additionally, experiments have been conducted on CIFAR$^{20}$. This dataset consists of 50,000 train images and 10,000 test images. All images are $32 \times 32$ pixels in size. CIFAR has two different sets of labels, CIFAR-100 and CIFAR-10. The following experiments have been conducted on CIFAR-10, where the images are divided into ten different classes. CIFAR is more challenging than MNIST from an unsupervised perspective. The images in CIFAR contain far more noise outside of the object to be clustered as is the case in MNIST. As such, this dataset allows us to draw stronger conclusions about the effectiveness of ACOL and Ordered ACOL-PL than would have been possible on MNIST alone. Examples of the CIFAR dataset are shown in Figure 4.2.

### 4.2 Superclasses and Subclasses

Unless mentioned otherwise, the following superclasses were used. The ten MNIST classes 0-9 were divided into five subclasses, zero through four, for the first superclass; and five subclasses, five through nine, for the second superclass. The ten classes of CIFAR-10 were divided into superclasses in a similar fashion: all
man-made classes form the subclasses of the first superclass, and all animals form the subclasses for the other. This resulted in two superclasses, one with four subclasses: ‘Car’, ‘Truck’, ‘Airplane’, and ‘Ship’, and one superclass with six subclasses: ‘Frog’, ‘Cat’, ‘Dog’, ‘Horse’, ‘Deer’, and ‘Bird’. These are informative superclass definitions, because they contain some information about their subclasses. In case a superclass definition is used where this is not the case, it will be referred to as ‘uninformative’. An example of such a superclass definition is to horizontally flip all images, and use ‘flipped’ and ‘not flipped’ as the superclasses.

4.3 Metrics

The conducted experiments will often require a comparison to be made between an ACOL network, which creates a clustering, and a classification network, which provides a predicted label. Assessing the quality of a classifier is normally done using a classification accuracy, such a classification accuracy cannot be directly applied to a clustering. As such, for ACOL, ACOL-PL and Ordered
ACOL-PL the quality is assessed using the purity of the clusters, and this is compared to the accuracy of the classification networks. The definition of the purity metric is shown in Equation 4.1.

\[
\text{Purity} = \frac{1}{N} \sum_{i=1}^{k} \max_{j} |C_i \cap K_j|
\]  

(4.1)

Where \(N\) is the total number of instances in the test set, \(k\) is the number of classes, \(C\) is the set of clusters and \(K\) the set of classes. It is the average of the total number of instances that match the label occurring most often in that cluster. This metric can safely be compared to the accuracy of the classifiers, as the accuracy metric used for regular classification networks corresponds directly to this purity metric. The definition of the accuracy metric is shown in Equation 4.2.

\[
\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{k} |M_i \cap K_i|
\]  

(4.2)

Where \(N\) is again the total number of instances in the test set, \(k\) is the number of classes, \(M\) is the set of nodes and \(K\) the set of classes. It is the average of the total number of instances that match the label of the node they are assigned to. As both the purity and the accuracy metric say something about accuracy with which the subclasses are placed in a cluster, throughout this thesis both metrics will be referred to as the “accuracy”. Note that, in case this reported accuracy is the result of a clustering, which is the case for all the ACOL baselines, the metric used is in fact the purity, and when the accuracy is the result of direct classification, the actual accuracy metric is used.

One caveat in using the purity metric is that the labels for each node are assigned based on the test data. This could positively effect the performance of the network, as the most optimal assignment of labels is always used. In the discussion (Section 5) this point is addressed further and it is shown that this has no significant influence.
4.4 Baselines

For the majority of the experiments, Ordered ACOL-PL is compared to four different baselines.

The first baseline is ACOL-PL. This baseline consists of the same network with the same loss function (shown in Equation 3.7) as Ordered ACOL-PL. The difference is that ACOL-PL is trained on all available data from the start, rather than first only on the labelled portion, followed by training on everything. A comparison with this baseline will show the value of the two-staged approach.

A comparison to ACOL\textsuperscript{16} is made to show the benefit of the included subclass label in Ordered ACOL-PL. ACOL is always trained with only the superclass labels provided, as it is unable to use subclass labels. The general architecture of this network is shown in Figure 4.3a and Figure 4.3b.

A third baseline consists of direct classification using a normal neural network. This network does not have the $2 \times 5$ subclass layer, but rather ends in a normal softmax layer with a node for each superclass or subclass it classifies to. Its architecture is illustrated in Figure 4.3c. Direct classification is always only trained on the percentage of labelled data, as it is unable to use unlabelled data.

A fourth baseline is Ordered ACOL-PL stage 1, as the name suggests, this is the first stage in training Ordered ACOL, where the network has been trained on just the labelled data points. The proposed method, Ordered ACOL-PL, will be referred to as Ordered ACOL-PL stage 2.

A final baseline that is used is noACOL. It has the same architecture as Ordered ACOL-PL, ACOL-PL and ACOL, as seen in Figure 4.3a and Table 3.1. Its loss function does include the subclass cross-entropy, applied on the Matrix softmax layer, but it lacks the balance, affinity and coactivity functions that cause ACOL to cluster. Equation 4.3 shows the final loss function noACOL uses. The noACOL network is trained with the same data as ACOL-PL, as it is able to use both the superclass labels and the subclass labels.

$$loss = c_0 \gamma_{\text{superclass}} + c_0 \gamma_{\text{subclass}} + c_4 \|Z\|_F^2$$

(4.3)
4.5 Experiment 1 - Training on partially labelled data

This experiment shows the effectiveness of our proposed Ordered ACOL-PL method. In order to see how Ordered ACOL-PL compares to the basic ACOL and to a regular neural network, these models are evaluated under a decreasing number of labelled instances. This experiment was conducted on both MNIST and CIFAR-10.

Our proposed method, Ordered ACOL-PL, is compared to the baselines. Because ACOL is not designed to take any labelled subclasses into account, the accuracies shown are not dependent on the varying percentage of labelled instances provided. We hypothesise that Ordered ACOL-PL will perform similar to direct classification when many labels are available, and that it will outperform all baselines when few labels are available. This would result in Ordered ACOL-PL being the correct choice regardless of the number of labelled data.

4.5.1 Results on MNIST

Table 4.2 shows the classification accuracies on the subclasses for MNIST. This data is also illustrated in Figure 4.4. On MNIST, Ordered ACOL-PL shows a
clear benefit over the other applied methods, as it consistently obtains the highest accuracy. Significance across ten runs is shown in Table 4.3 for 1% of the data labelled. The significance is calculated with the Wilcoxon signed-rank test, on the default MNIST test set, to allow for easy comparison with other methods. The Wilcoxon signed-rank test was chosen as all assumptions are met. The dependent variable is continuous, two independent runs on the same test set are compared each time and the distribution of differences is symmetric, insofar that can be seen using ten samples.

**Table 4.2:** Subclass classification accuracy on MNIST. The highest accuracy per percentage labelled is indicated in bold.

<table>
<thead>
<tr>
<th>Labelled %</th>
<th>Ordered ACOL-PL stage 2</th>
<th>Ordered ACOL-PL stage 1</th>
<th>ACOL-PL</th>
<th>ACOL Direct to subclass</th>
<th>noACOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td><strong>0.989</strong></td>
<td><strong>0.989</strong></td>
<td>0.805</td>
<td>0.983</td>
<td>0.985</td>
</tr>
<tr>
<td>10</td>
<td><strong>0.983</strong></td>
<td>0.975</td>
<td>0.981</td>
<td>0.805</td>
<td>0.956</td>
</tr>
<tr>
<td>3</td>
<td><strong>0.981</strong></td>
<td>0.956</td>
<td>0.921</td>
<td>0.805</td>
<td>0.950</td>
</tr>
<tr>
<td>1</td>
<td><strong>0.978</strong></td>
<td>0.932</td>
<td>0.834</td>
<td>0.805</td>
<td>0.894</td>
</tr>
<tr>
<td>0.1</td>
<td><strong>0.948</strong></td>
<td>0.716</td>
<td>0.822</td>
<td>0.805</td>
<td>0.665</td>
</tr>
</tbody>
</table>
For a large amount of labelled data, one could use either direct classification or Ordered ACOL-PL. However, when the portion of labelled data decreases, Ordered ACOL-PL performs significantly better than all other baselines with 1% of data labelled (at $p < 0.01$). The poor performance for noACOL on lower
percentages of labelled data clearly shows that the ACOL loss is an important factor in the good performance of Ordered ACOL-PL. The accuracy of noACOL is not reported on CIFAR, because these results on MNIST more than suffice to draw the conclusion that the performance of ACOL is not merely explained by its architecture, but that the loss is a vital component.

Table 4.2 shows that indeed Ordered ACOL-PL shows equal or better performance than the baselines, regardless of the amount of labelled data. It also shows that the direct classification does not suffer a lot under the small subset of the data it is trained on, even when only 0.1% of the data is labelled. At that point the network is only trained on roughly five images per class (0.1% of 45000 is 45 images in total, resulting in 4.5 images per class), and the network still generalises surprisingly well to the test set. A comparison of this direct classification to the first training stage of Ordered ACOL-PL (Ordered ACOL-PL stage 1), which are both only trained using the labelled data, shows that in this case using ACOL might already provide some benefit over training without ACOL, for 0.1%. After the second stage of training, Ordered ACOL-PL (Ordered ACOL-PL stage 2) clearly performs best.

The loss curve of Ordered ACOL-PL with 0.1% of the data labelled, shown in Figure 4.5, shows that this is because the second stage of learning allows for much better generalisation to the test set. For the first 30,000 epochs (stage 1) the network clearly has difficulty generalising to the unseen validation set, despite being able to classify the train set quite accurately. Once the other train data is added in stage 2, the validation loss and the train loss merge, indicating that now the network is able to properly generalise to unseen data.
4.5.2 Results on CIFAR

Table 4.4 and Figure 4.6 show the results on the CIFAR dataset. On CIFAR, it is also observed that the Ordered ACOL-PL accuracy is mostly higher than the accuracy of ACOL, only with 0.1% labelled the seed learned during the first stage is too weak to beat ACOL. The direct classification outperforms Ordered ACOL-PL for lower percentages of labelled data. This behaviour shows that Ordered ACOL-PL is not necessarily an improvement if the base ACOL performs poorly. Where ACOL on MNIST obtained a subclass accuracy of 88.4%, ACOL on CIFAR only obtains a subclass accuracy of 51.0%. On MNIST, the performance of Ordered ACOL-PL slightly improves when using this clustering, whereas it appears that on CIFAR the clustering works against the information learned from the labelled subclasses. An additional factor could be that the first stage of Ordered ACOL learns a stronger seed on MNIST than it does on CIFAR (67.9% on CIFAR and 93.2% on MNIST). This could be caused by the fact that the dataset is harder to learn with less data, as also indicated by the lower direct classification accuracy on CIFAR (65.3% on 1% labelled) compared to MNIST (89.4% on 1% labelled).

With 1% of the data labelled Ordered ACOL-PL significantly outperforms (at
all other baselines on CIFAR, with exception of direct classification. The significance was determined using the Wilcoxon signed rank test, because the assumptions for this test are mostly met. The dependent variable is continuous, the independent variable is compared across two related groups, as the test sets are equal, and finally the differences between each group are symmetrically distributed, although with the used sample size of five runs per method this is hard to tell. The results of this Wilcoxon test, comparing all the Ordered ACOL-PL accuracies on 1% of labelled data, is shown in Table 4.5.

Table 4.4: Subclass classification accuracy on CIFAR-10. The highest accuracy per percentage labelled is indicated in bold.

<table>
<thead>
<tr>
<th>Labelled %</th>
<th>Ordered ACOL-PL stage 2</th>
<th>Ordered ACOL-PL stage 1</th>
<th>ACOL-PL</th>
<th>ACOL</th>
<th>Direct to subclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.940</td>
<td>0.940</td>
<td>0.940</td>
<td>0.516</td>
<td>0.866</td>
</tr>
<tr>
<td>10</td>
<td>0.878</td>
<td>0.849</td>
<td>0.781</td>
<td>0.516</td>
<td>0.872</td>
</tr>
<tr>
<td>3</td>
<td>0.741</td>
<td>0.679</td>
<td>0.780</td>
<td>0.516</td>
<td>0.821</td>
</tr>
<tr>
<td>1</td>
<td>0.726</td>
<td>0.679</td>
<td>0.600</td>
<td>0.516</td>
<td>0.752</td>
</tr>
<tr>
<td>0.1</td>
<td>0.483</td>
<td>0.325</td>
<td>0.543</td>
<td>0.516</td>
<td>0.564</td>
</tr>
</tbody>
</table>
Figure 4.6: Subclass classification accuracy on CIFAR of different methods under varying percentages of labelled data.

Table 4.5: Mean and standard deviation of the subclass accuracy of five runs for all tested methods on 1% of labelled data on CIFAR. Also, the Wilcoxon p-value of each method compared to Ordered ACOL-PL stage 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Std.</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered ACOL-PL stage 2</td>
<td>0.726</td>
<td>0.052</td>
<td>-</td>
</tr>
<tr>
<td>Ordered ACOL-PL stage 1</td>
<td>0.679</td>
<td>0.015</td>
<td>0.036</td>
</tr>
<tr>
<td>ACOL-PL</td>
<td>0.600</td>
<td>0.042</td>
<td>0.018</td>
</tr>
<tr>
<td>ACOL</td>
<td>0.516</td>
<td>0.013</td>
<td>0.043</td>
</tr>
<tr>
<td>Direct to subclass</td>
<td>0.752</td>
<td>0.007</td>
<td>0.080</td>
</tr>
</tbody>
</table>

The hypothesis for this experiment can be mostly confirmed; Ordered ACOL-PL will perform similar to direct classification when many labels are available, and it will outperform all baselines on MNIST when few labels are available. On CIFAR direct classification to the subclasses obtains a slightly higher accuracy than Ordered ACOL-PL. The experiments that follow will further explain this difference in performance.
4.6 Experiment 2 - Further exploration of relation between ACOL and Ordered ACOL-PL using uninformative superclasses

One of the conclusions reached in the previous experiment is that the accuracy of Ordered ACOL-PL depends on the performance of ACOL. Ordered ACOL-PL was outperformed by direct classification on CIFAR, whereas on MNIST it achieved an accuracy above that of direct classification and all other baselines. Ordered ACOL-PL is also heavily dependent on the strength of the seed learned during the first stage of training, as seen in the previous experiment in Table 4.4. With only 0.1% labelled on CIFAR the first stage clearly struggles to learn a representative seed, and this causes the accuracy of Ordered ACOL-PL to even go below the accuracy of ACOL. A more clear indication of the dependency of Ordered ACOL-PL on ACOL is that the unordered variant, ACOL-PL, performs equal to ACOL for low percentages of labelled data. Ordered ACOL-PL does not follow this trend precisely, as a strong seed generally contributes to around the direct classification level. But if ACOL performs poorly, it will still end up clustering images together on undesired properties. In Experiment 1 this dependency of Ordered ACOL-PL on ACOL was shown for two informative superclasses. This experiment tests whether this is also the case when an uninformative superclass is used. We hypothesise that the subclass accuracy of Ordered ACOL-PL is dependent on ACOL for few labels on an uninformative superclass definition as well. This new, uninformative, superclass definition is ‘flipped’ versus ‘not flipped’; each instance is provided to the network either as is, or flipped horizontally. The superclass labels ‘flipped’ and ‘not flipped’ are assigned according to whether or not the image has been flipped. Each superclass is then clustered into ten different clusters. The accuracy is presented against the baselines in Table 4.6 and Figure 4.7.
Table 4.6: The subclass classification accuracy on MNIST with ‘flipped’ vs. ‘not flipped’ as super-classes, of different methods under varying percentages of labelled data. The highest accuracy per percentage labelled is indicated in bold.

<table>
<thead>
<tr>
<th>Labelled %</th>
<th>Ordered ACOL-PL stage 2</th>
<th>Ordered ACOL-PL stage 1</th>
<th>ACOL-PL</th>
<th>ACOL</th>
<th>noACOL</th>
<th>Direct to subclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.977</td>
<td>0.977</td>
<td>0.977</td>
<td>0.578</td>
<td>0.962</td>
<td>0.983</td>
</tr>
<tr>
<td>10</td>
<td>0.965</td>
<td>0.965</td>
<td>0.920</td>
<td>0.578</td>
<td>0.112</td>
<td>0.956</td>
</tr>
<tr>
<td>1</td>
<td>0.954</td>
<td>0.938</td>
<td>0.649</td>
<td>0.578</td>
<td>0.112</td>
<td>0.894</td>
</tr>
<tr>
<td>0.1</td>
<td>0.809</td>
<td>0.774</td>
<td>0.596</td>
<td>0.578</td>
<td>0.113</td>
<td>0.665</td>
</tr>
</tbody>
</table>

It can be concluded that the choice of superclass influences the performance of ACOL quite strongly. Switching from the informative superclass definition to the uninformative one decreases the accuracy of ACOL from 88.4% to 65.1%. Ordered ACOL-PL still outperforms the direct classification, although under this uninformative superclass the accuracy is only 80.9% for 0.1% of labelled data.
data, against an accuracy of 94.8% for the informative superclass in Table 4.2. Even though a poorer ACOL performance does hurt the performance of Ordered ACOL-PL, the seed it learns on MNIST is strong enough to still outperform direct classification. This is unlike the results on CIFAR in Table 4.4 in the previous experiment, where it appears to be more challenging to learn the subclasses from the small amount of labelled data, and thus Ordered ACOL-PL does not have as strong a seed to cluster around in the second stage.

This data shows that, in line with Experiment 1, ACOL-PL closely follows ACOL for lower percentages of labelled data. The performance of Ordered ACOL-PL, although still better than other baselines, is not as good as under the informative superclass definition. From this, our hypothesis is confirmed and it can be concluded that indeed the ability of ACOL to form clusters that correlate well with the desired classes is crucial for the performance of Ordered ACOL-PL.

Noticeable is that noACOL performs far worse than it did under the previous superclass definition. This is unsurprising because in this setting noACOL is not aided in classifying the subclasses by the superclass information. It has to learn to classify within each superclass between ten classes rather than five. Where this increase in subclasses to classify is a problem for noACOL, the influence of this factor on the other methods is clearly not as strong. Based on the data in Table 4.6, it can be concluded that other methods still perform reasonably well and are not, or only marginally, effected by this increase in the number of subclasses per superclass. The next experiment will show that indeed the number of subclasses does not explain the decrease in the performance of ACOL and (Ordered) ACOL-PL.

4.7 EXPERIMENT 3 - VARYING NUMBER OF SUBCLASSES

Aside from the superclasses used, another variable in most experiments conducted is the number of subclasses to be classified within one superclass. For the informative superclass definition on MNIST, only five subclasses need to be classified per superclass, as opposed to the ten subclasses under the uninfor-
tive superclass. We hypothesise that the number of subclasses ACOL needs to classify per superclass does not explain the difference between different superclass definitions with a varying number of subclasses. To exclude the possibility that this is a major factor in the difference in performance between the superclasses, the number of subclasses is varied for the uninformative superclass definition of MNIST. Where normally this superclass definition has ten subclasses per superclass, now some of the subclasses will be removed. The subclass sets 5 through 9 (5 subclasses), 0 through 4 (5 subclasses) and 0 through 9 (10 subclasses) are tested. All conducted experiments always use a number of clusters equal to the number of subclasses, as this is required by the ACOL-PL methods. This experiment therefore also uses as many clusters as there are subclasses in each run. The resulting subclass accuracies of ACOL are reported in Table 4.7.

Table 4.7: The classification accuracies of ACOL on MNIST, under the uninformative superclasses using a varying number of subclasses.

<table>
<thead>
<tr>
<th>Included subclasses</th>
<th>Subclass</th>
<th>Superclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-9</td>
<td>0.335</td>
<td>0.974</td>
</tr>
<tr>
<td>0-4</td>
<td>0.928</td>
<td>0.989</td>
</tr>
<tr>
<td>0-9</td>
<td>0.578</td>
<td>0.984</td>
</tr>
</tbody>
</table>

These results show that less subclasses to classify per superclass do not necessarily lead to an increase in accuracy. Five subclasses (5-9) per superclass under the uninformative superclass definition are only classified correctly with an accuracy of 33.5%, whereas the informative superclass definition also has five subclasses per superclass and performs significantly better at an accuracy of 80.5%. The accuracy does vary strongly over the different subsets of subclasses selected. To illustrate this further, if the subclasses 0 through 4 are used instead of 5 through 9, an accuracy of 92.8% is reached. This means that it might not be possible to find a single superclass definition that is optimal regardless of the dataset. But instead each dataset, with each their own subclasses, might have a different optimal superclass definition.
4.8 Experiment 4 - Superclass difficulty

To explain the performance differences between Ordered ACOL-PL with the informative superclass on MNIST in Table 4.2 and Ordered ACOL-PL with the uninformative superclass definition in Table 4.6, the influence of the different supersclasses on the accuracy of ACOL needs to be inspected. The informative superclass definition leads ACOL to a subclass accuracy of 88.4%, whereas the uninformative superclass leads to an accuracy of a mere 57.8%. As ACOL clusters on the Fc7 layer of the network, the final features before classification, the features that the network learns for this layer are vital in order to obtain a desired clustering. Apparently the features ACOL clusters on are much more useful in the informative setting, than in the uninformative one. We hypothesise that the informative superclass definition requires more knowledge about the individual classes than the uninformative superclass definition. If this explicit information about the subclasses is indeed required for the correct classification of the informative supersclasses, the Fc7 layer would contain this explicit knowledge. This is then an explanation for the difference in the performance of ACOL.

To test the hypothesis, the direct classification to the supersclasses is compared to the direct classification of the subclasses on the MNIST dataset. When the superclass classification accuracy is dependent on the subclass classification accuracy it can be concluded that the superclass requires knowledge of the subclasses to be learned. When this is not the case it can be concluded that in order to classify the supersclasses correctly, no knowledge of the subclasses is needed. The results are shown in Table 4.9 and Figure 4.8.

Table 4.8: The classification accuracy of direct classification on MNIST. The highest accuracy per percentage labelled is indicated in bold.

<table>
<thead>
<tr>
<th>Labelled %</th>
<th>Subclass</th>
<th>Informative superclass</th>
<th>Uninformative superclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.983</td>
<td>0.978</td>
<td>0.994</td>
</tr>
<tr>
<td>10</td>
<td>0.956</td>
<td>0.969</td>
<td>0.994</td>
</tr>
<tr>
<td>1</td>
<td>0.894</td>
<td>0.934</td>
<td>0.994</td>
</tr>
<tr>
<td>0.1</td>
<td>0.665</td>
<td>0.800</td>
<td>0.994</td>
</tr>
</tbody>
</table>
Figure 4.8: The superclass classification accuracy of the direct classification networks under varying percentages of labelled MNIST data.

The direct classification to the uninformative superclass is very minimally affected by the decrease in the amount of data, whereas the direct classification to the informative superclass becomes less accurate with fewer training data supplied. We hypothesised that the classification accuracy of the informative superclass is depended on the performance of the subclasses, and indeed, the drop in accuracy for the informative superclass classification is strongly correlated with the drop in accuracy for the classification of the subclasses, showing that the informative superclass classification task is depended on knowledge of the individual subclasses. Therefore it can be concluded that our hypothesis is correct; the informative superclass definition requires explicit knowledge of the subclasses. This is then an example of how the superclass definition can steer the quality of the clustering produced by ACOL; a superclass definition that relies on knowledge of the individual subclasses or otherwise guides the network to learn features that are informative for a desired clustering, results in a better ACOL subclass accuracy than a superclass that does not. As seen previously in Experiment 1 and Experiment 2: when comparing ACOL on MNIST under
the informative and uninformative superclasses, ACOL produces more desirable clusters in the informative setting than it does in the uninformative setting.

An explanation for these results is that informative superclasses consist of a grouping of visually unrelated classes, meaning there is no clear single visual characteristic on which the digits 0 through 4 and 5 through 9 can be separated. The uninformative superclass definition does not do this, but rather sets up a classification problem between two groups of visually strongly related images. All handwritten digits will tend to lean to the right. This means that in one superclass, the average digit leans right, and in the other superclass, the average digit will lean left. This effect is clearly visible in Figure 4.9. As a result, the features that are learned for ACOL to cluster on with the informative superclasses are more relevant to cluster into the desired subclasses, than is the case with the uninformative superclasses.

![Figure 4.9: Average digit in the informative and uninformative MNIST superclasses.](image)

4.9 EXPERIMENT 5 - SUPERCLASS DEFINITIONS

As shown in Experiment 1 and Experiment 2, the superclass used has a strong influence on the performance of ACOL, and thus of Ordered ACOL-PL. In Experiment 4 it was argued that a good superclass definition is one that relies on knowledge of the individual subclasses or otherwise guides the network to learn features that are informative for a desired clustering. This fifth experiment ex-
plores the effects of various superclass definitions, and it attempts to find a superclass definition that leads to an improved subclass classification accuracy of ACOL on CIFAR. We hypothesise that the superclass definition largely determines the type of clusters formed by ACOL.

Thus far, two types of superclasses have been identified: informative and uninformative superclasses. The first group requires some sort of knowledge about the content of the image, and therefore still requires some level of annotation effort on the side of the user. Uninformative superclasses, such as whether the image is flipped or not, can be annotated automatically by the computer, without any effort by the user. The informative superclasses are therefore undesirable and using the uninformative superclasses, that do not require annotation effort, would be preferable. However, the informative superclasses lead to the best performance.

First, three informative superclass definitions are tested. The first of these being four man-made classes in one superclass, and six animals in the other, this superclass definition has been used thus far. In addition to this it is also tested whether the balance of these superclasses forms a problem. To accomplish this the ‘horse’ is moved from the animals superclass to the man-made superclass. Other than the fact that the man-made superclass contains only vehicles, and one could argue a horse is a vehicle, this pick was arbitrary. The final informative superclass definition tested is one where the man-made and animal classes are mixed. This way two superclasses are created that are visually as different as possible. This follows the reasoning in Experiment 4, where it was concluded that superclasses that lead to improved ACOL results are hard to visually distinguish from each other as a whole, but are easier to distinguish when treated as a combination of their respective subclasses. These mixed superclasses then become a combination of ‘Airplane’, ‘Automobile’, ‘Cat’, ‘Deer’, ‘Frog’ for the first superclass and a combination of ‘Bird’, ‘Dog’, ‘Horse’, ‘Ship’ and ‘Truck’ for the second superclass. An effort was made to place much confused classes, such as ‘Bird’ and ‘Airplane’ in different superclasses.

In addition to these three informative superclasses, four uninformative superclasses are tested. First, the superclasses flipped and not flipped are tested, to
mirror the uninformative superclass used on MNIST.

Another cheap to annotate superclass definition tested is one of little blue versus a large amount of blue, each with ten subclasses. To create this superclass definition, each image is placed one of two superclasses, where the median average blue value acts as the threshold. It is expected that the clusters that result from this superclass definition are primarily based on the average amount of blue in each image. Even though this will therefore not produce a desired clustering into the respective subclasses, it will give additional insight in how the superclass definition steers the clustering.

Finally, two pairs of uninformed superclasses are defined based on the features generated by a VGG-16 network pretrained on ImageNet, one on the Fc7 layer, and one on the final convolutional layer (Pool5). The expectation is that these features contain some information about the content of the image. By drawing the decision boundary for the superclass classification in this space, the expectation is that the network would first also have to learn a feature representation similar to the VGG16-layer, which would then contain similar information about the content of the images. A clustering on this learned feature space should then be based on the contents of the images, and potentially provide a desired clustering. The decision boundary between the two superclasses is drawn by letting k-means find two clusters.

The performance of ACOL under these different superclass definitions is shown in Table 4.9
Table 4.9: The classification accuracy of ACOL on a variety of superclass definitions. The highest accuracies are indicated in bold.

<table>
<thead>
<tr>
<th>Superclass</th>
<th>Subclass Accuracy</th>
<th>Superclass Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>man-made and animals</td>
<td><strong>0.510</strong></td>
<td><strong>0.990</strong></td>
</tr>
<tr>
<td>man-made and animals (balanced)</td>
<td>0.504</td>
<td>0.972</td>
</tr>
<tr>
<td>mixed man-made and animals</td>
<td>0.398</td>
<td>0.788</td>
</tr>
<tr>
<td>Fc7</td>
<td>0.417</td>
<td>0.486</td>
</tr>
<tr>
<td>Pool5</td>
<td>0.469</td>
<td>0.545</td>
</tr>
<tr>
<td>Threshold blue</td>
<td>0.279</td>
<td>0.526</td>
</tr>
<tr>
<td>Flipped vs not flipped</td>
<td>0.316</td>
<td>0.511</td>
</tr>
</tbody>
</table>

The results show that, like on MNIST, the informative superclasses outperform the uninformative ones. Interesting is that the superclass definition that is visually more easily separable, man-made and animals, outperform the mixed man-made and animals superclasses, which would require more distinct subclass knowledge to accurately classify the superclass. All the uninformative superclasses are apparently hard to classify, which likely also explains their poor subclass accuracy. Noteworthy is also that the performance is not increased through the use of a balanced superclass definition.

As expected, the poorest subclass accuracy is obtained by the superclasses determined by a threshold on the average blue in an image. This is in part caused by the fact that the convolutional layers are pretrained on ImageNet and frozen, and it is therefore unlikely that much colour information is retained. Even though the superclass accuracy is just above random, in the clustering produced by the network it is visible, with some good will, that the network has been trained to look at the average blue value. The superclass with images that are more blue than the average contains more white, blue and purple images. Whereas the superclass with images that are less blue than the average contain more black, red and green images. Examples of some of the found clusters using this superclass are shown in Figure 4.10.

The superclasses used clearly have a large influence on the features learned and the performance of ACOL. The difference in sub- and superclass accuracies
between the different superclasses are large. The superclasses defined on the
colour of the image also clearly result in clusterings based on the average colour.
This confirms our hypothesis that the superclass definition largely determines
the type of clusters formed by ACOL.

4.10 Experiment 6 - Altering the clustering space - ACOL

Experiment 4 and Experiment 5 already established that the used superclass
definition controls the features learned by the network going into ACOL. With
better, more relevant, features the subclass accuracy of ACOL is increased, and
therefore the Ordered ACOL-PL performance is increased as well. Where the
superclasses influence this clustering space from a “top down” perspective, the
features that go into the ACOL layer can also be influenced using a “bottom up”
approach. As clustering is generally hurt by a high-dimensional feature space,
one would expect the performance of ACOL to improve with fewer features to
cluster on. But since the clustering in ACOL is based on a weighted set of fea-
tures, we hypothesise that ACOL is capable of selecting the features relevant
for the desired clustering. To test this hypothesis, this experiment will exam-
ine various handmade (rather than learned) selections of the features going into
the ACOL layer for the CIFAR network. When no clear improvement is seen, or
when a decrease in accuracy is witnessed, it can be concluded that our hypothe-
sis is true.

For all previous experiments on CIFAR the ACOL layer was placed on the
Fc7 layer of VGG16, this will therefore be the primary baseline here (Fc7). The
CIFAR input is scaled and transformed to match the ImageNet parameters with which the VGG16 network was trained originally. The first way to reduce the number of features is to cluster on the Fc8 layer of the VGG16 network. The Fc8 layer is the final fully connected layer in VGG16 and contains a 1000 features, corresponding to a 1000 ImageNet classes, as opposed to the 4096 features in the Fc7 layer. This space is further reduced by manually selecting all features from the Fc8 layer that correspond logically to all ImageNet classes that are relevant to the 10 classes in CIFAR (Fc8[related classes]). A total of 257 ImageNet classes were found to be logically related to the CIFAR classes.

The number of features is also reduced by performing feature selection based on the variance within that feature. It is common practice to apply PCA to a feature space before clustering, because the classes are easier to separate in a feature space with high variance. From the Fc7 layer both the 2048 features with the highest variance (Fc7[2048]), as well as the top 400 features with the highest variance (Fc7[400]) are selected. The ACOL layer on the MNIST network is also supplied with 2048 features. The variances were determined on the output of the Fc7 layer (after the ReLU) of the VGG16 network trained on ImageNet when given the CIFAR trainset as input. The results are shown in Table 4.12.

<table>
<thead>
<tr>
<th>Feature space</th>
<th>Subclass Accuracy</th>
<th>Superclass Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fc7</td>
<td><strong>0.510</strong></td>
<td><strong>0.992</strong></td>
</tr>
<tr>
<td>Fc8</td>
<td>0.488</td>
<td>0.990</td>
</tr>
<tr>
<td>Fc8[related classes]</td>
<td>0.485</td>
<td>0.990</td>
</tr>
<tr>
<td>Fc7[2048]</td>
<td>0.497</td>
<td>0.990</td>
</tr>
<tr>
<td>Fc7[400]</td>
<td>0.489</td>
<td>0.983</td>
</tr>
</tbody>
</table>

Table 4.10 shows that reducing the number of features does not improve the performance of ACOL, but instead slightly decreases it. For example, using the Fc8 layer of VGG16 did not provide an increase in the subclass accuracy of ACOL. Furthermore, selecting only the related classes from this layer does
not provide an increase in the accuracy, but rather decreases it slightly. The same holds for the selections made on the Fc7 layers, which only decrease the accuracy as compared to the full Fc7 layer. These results confirm the hypothesis that ACOL is already quite capable of selecting the features useful for the clustering. Reducing the feature space ACOL clusters in, with the approaches used in this experiment, is not an effective method to improve clustering performance.

4.11 Experiment 7 - Altering the cluster space - Ordered ACOL-PL

The previous experiment focussed on increasing the subclass accuracy of the base ACOL, and improve Ordered ACOL-PL that way. Ordered ACOL-PL does have one benefit in this regard over ACOL; it has labels for some portion of the data available. This data could be used to select or learn more descriptive features of the classes. Nevertheless, we hypothesise that Ordered ACOL-PL, despite having additional information available for the feature selection, also is not aided by manual feature selection.

For this experiment, like in the previous experiment, the unaltered Fc7 layer is used as the baseline, this is the variant used throughout this thesis. Additionally, three variations are considered.

Since the direct classification still outperforms Ordered ACOL-PL on CIFAR despite having seen less data, the information that the direct network learns should be used to learn a more accurate clustering with Ordered ACOL-PL. To be precise, the ACOL loss is removed in the first stage of Ordered ACOL-PL, where the ACOL-PL network is trained on only the labelled subclasses. What remains is only the cross-entropy loss over the subclasses, which is effectively equal to the direct classification network (Direct 1%). This variant on the first stage could provide a stronger seed to cluster around than the regular first stage of Ordered ACOL-PL. Because in Table 4.4, direct classification outperforms this first stage for lower percentages of labelled data. This is only true for Ordered ACOL-PL on CIFAR however, as for MNIST the first stage of Ordered ACOL-PL outperforms direct classification for 1%, as can be seen in Table 4.2.
For the second variation, a selection of the feature space is made based on the one or two classes from the Fc8 layer of the ImageNet-trained VGG16 network which correlate strongest with a class in CIFAR (Fc8[10] and Fc8[20]). Meaning that if the VGG16 network has an ‘Airliner’ class that fires strongest whenever one of the CIFAR images of which it is known that the label is ‘Airplane’ is provided, the ‘Airliner’ class is included in the feature selection. This selection is done for each of the 10 CIFAR classes. The ImageNet classes selected for each of the CIFAR10 classes are shown in Table 4.11. The results are shown in Table 4.12.

**Table 4.11:** The VGG16 classes selected for each of the CIFAR10 classes.

<table>
<thead>
<tr>
<th>CIFAR 10</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane</td>
<td>Airliner</td>
</tr>
<tr>
<td>Automobile</td>
<td>Moving van</td>
</tr>
<tr>
<td>Bird</td>
<td>Black grouse</td>
</tr>
<tr>
<td>Cat</td>
<td>Milk can</td>
</tr>
<tr>
<td>Deer</td>
<td>Hartebeest</td>
</tr>
<tr>
<td>Dog</td>
<td>Japanese spaniel</td>
</tr>
<tr>
<td>Frog</td>
<td>Fox squirrel</td>
</tr>
<tr>
<td>Horse</td>
<td>Sorrel</td>
</tr>
<tr>
<td>Ship</td>
<td>Container ship</td>
</tr>
<tr>
<td>Truck</td>
<td>Moving van</td>
</tr>
</tbody>
</table>

Table 4.12: The subclass and superclass classification accuracies of Ordered ACOL-PL on CIFAR using a variety of feature selections. The highest accuracies are indicated in bold.

<table>
<thead>
<tr>
<th>Feature space</th>
<th>Stage 1 Subclass</th>
<th>Stage 1 Superclass</th>
<th>Stage 2 Subclass</th>
<th>Stage 2 Superclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fc7</td>
<td>0.679</td>
<td>0.940</td>
<td>0.726</td>
<td>0.987</td>
</tr>
<tr>
<td>Direct 1%</td>
<td>0.220</td>
<td>0.508</td>
<td>0.24375</td>
<td>0.526</td>
</tr>
<tr>
<td>Fc8[10]</td>
<td>0.234</td>
<td>0.531</td>
<td>0.243</td>
<td>0.524</td>
</tr>
<tr>
<td>Fc8[20]</td>
<td>0.313</td>
<td>0.493</td>
<td>0.326</td>
<td>0.741</td>
</tr>
</tbody>
</table>
These results confirm our hypothesis that Ordered ACOL-PL, like ACOL in the previous experiment, is able to select the features in a way that leads to a good feature space for clustering, and that it is therefore not aided by manual feature selection. Limiting the number of features that Ordered ACOL-PL can use still does not improve the accuracy, despite the additional knowledge used in comparison to the previous experiment.
The conducted experiments have proven Ordered ACOL-PL as a promising technique that, given that a strong superclass definition is found, can achieve a significant increase in performance for datasets with few subclass labels. It was shown that this is because Ordered ACOL-PL first learns a strong seed during the first training stage where it is provided with the limited amount of labelled data. After this first stage the network is slightly overtrained on the small train set, and the performance on unseen validation data is relatively low. This is remedied during the second training stage, where Ordered ACOL-PL clusters the unlabelled data around these previously learned data points from stage 1. This way ACOL is given a strong seed for the desired clustering, and the overall method is able to use the unlabelled data to classify the subclasses, rather than forming any of the other possible clusterings. Some of the limitations of this work, and potential future work still to be done are discussed here.

The first limitation is that experiments are only conducted on MNIST and CIFAR, two relatively easy datasets. Because the performance of Ordered ACOL-PL is not consistent across these two datasets, it is hard to generalise to other datasets and predict whether it is the correct choice to use Ordered ACOL-PL. Future work should apply Ordered ACOL-PL on more complex datasets such
as ImageNet, MS-COCO or PASCAL VOC, to observe how its accuracy is influenced. More work should also be done to make Ordered ACOL-PL more robust under different combinations of superclasses and subclasses.

The second limitation to note is that the informative superclasses still require labelling of the entire dataset. However, with the exception of the informative superclass definition where the CIFAR classes are mixed, the labelling of the entire dataset would only require labelling with very coarse classes, and would still be easier to do than labelling with the specific class label. Although several uninformative superclasses, which do not require any labelling, were tested, no clear heuristic was found as to what kind of uninformative superclass definition leads to a good clustering performance.

A third limitation is that significance is not determined for all runs, because this was too costly considering the time available. However, since mostly the trend of the performance over the percentage labelled is considered, rather than the absolute accuracies, the conclusions are still valid, especially considering the generally large differences in accuracy. However it would have been desirable to compare the average of multiple runs, because dropout layers were used in the networks, and these do introduce fluctuations in the performance.

A fourth limitation is that the hyper parameters were not entirely optimised. The learning rate was tweaked, as well as the affinity threshold for larger number of clusters, and some experiments with the network architecture were conducted, but mostly the networks used for MNIST and CIFAR were used as-is. These could have been improved by for example using Leaky-ReLUs\textsuperscript{32} rather than the regular RELUs. Many other parameters could also have been optimised, such as the optimiser used, the number of layers and the width of the layers. However, the networks adapted already perform reasonably well on MNIST and CIFAR. Furthermore, because variations upon the networks are compared with each other, the absolute performance is not as important as the relative performance of the methods tested. The relative performance increase of Ordered ACOL-PL over other methods should carry over regardless of the underlying network architecture.

A fifth limitation is that for Experiment 6 and Experiment 7, no common fea-
ture selection algorithms, such as forward or backward selection have been used. However, the subsets of features that were tested already show that ACOL itself is capable of selecting the optimal set of features through the weights. Furthermore, even if a selection made through these algorithms could have improved the accuracy, it would be highly impractical to use. Since these algorithms generally require repeated evaluation of the accuracy with each new selection. Evaluating the accuracy of such a new feature selection would in this case require retraining the entire network using the new subset of features, and this would be very time consuming.

A final limitation lies with the metric used. For all clustering based methods the subclass classification accuracy was calculated using the purity, a common metric for clustering. The problem arises from the fact that the purity metric differs from a normal accuracy metric by the fact that a label is assigned to a cluster based on the test data. In other words, the purity metric always assigns the label to a cluster that results in the highest possible subclass accuracy for that test data. It would be more proper to assign the labels based on the 1% train data. This flaw was only detected after the fact, with little time to correct this mistake. Five runs of ACOL on the informative superclass for MNIST, measured using both the purity metric and the proper metric, showed that this way of using the metric does not result in a significant difference in performance. The purity metric resulted in an average accuracy of 79.5% and the more correct way of measuring resulted in an accuracy of 79.6%. The difference is not significant at a Wilcoxon signed-rank test p-value of 0.230. For this experiment five clusters per superclass were used. This is representative of all the experiments performed in this thesis, as in all cases the number of clusters is equal to the number of subclasses. From this small experiment it can therefore be concluded that this problem with the purity metric has not significantly affected the results in this thesis.

Another minor problem with the purity metric is how it scales in relation to the number of clusters. If, for instance, a network would have as many clusters as instances to classify, the accuracy could trivially go to 100% by simply assigning each instance its own cluster. Indeed, when experimenting with an
increasing number of clusters under a constant number of subclasses the accuracy increases, as shown in Table 5.1. Ordered ACOL-PL is bound to use as many clusters as there are subclasses, because each cluster is assigned a subclass label during the first stage of training. This increase in accuracy is therefore not easily exploited. Furthermore, Figure 5.1 shows that the additional clusters are not even fully exploited by ACOL. When the affinity is minimised the network is rewarded to maintain a strong boundary between the different clustering nodes, and the network therefore does not benefit from having many, hard to distinguish, clusters. Because throughout this thesis the number of clusters generally equal the number of subclasses, and the extra clusters are not even fully exploited by ACOL, this flaw in the purity metric has likely not affected the results in this thesis.

Table 5.1: The subclass classification accuracies (measured in purity) of ACOL on MNIST, under both informative and uninformative superclasses using a varying number of clusters.

<table>
<thead>
<tr>
<th>Number of clusters</th>
<th>Informative</th>
<th>Uninformative</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.884</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0.903</td>
<td>0.578</td>
</tr>
<tr>
<td>15</td>
<td>0.949</td>
<td>0.627</td>
</tr>
<tr>
<td>20</td>
<td>0.963</td>
<td>0.613</td>
</tr>
</tbody>
</table>

Figure 5.1: The average digit classified to each of the clusters under the informative superclass

Aside from these limitations, the work conducted here also has more potential, which for now will be left as future work. In this thesis the ACOL or (Ordered) ACOL-PL layer has always been placed after the non-linearity on the Fc7 layer of the network it used. At the time this was a sensible decision, because with any normal neural network you include a non-linearity between each layer. However, a later experiment revealed that the subclass classification accuracy of
ACOL is somewhat improved when it is placed directly on the Fc7 layer (From 51.0% to 53.3% on CIFAR). Because the non-linearity used in this case was a ReLU, this improvement can be explained by noting that apparently some of the features had negative values, and these were set to 0 by the non-linearity, resulting in the loss of some information. Placing ACOL directly on the Fc7 layer however, decreases the depth of the network, because two linear operations in a row should not result in any real benefit over a single linear operation. A potential experiment in the future would be to expand the softmax by adding an additional layer before each row in the superClass × subClass clustering layer, i.e. an extra layer specific to each superclass. This would give ACOL a layer that does not need to learn about the superclass, but instead is dedicated to optimising the feature space for clustering.

Another possible addition to Ordered ACOL-PL is to see how it handles a class for which it has no labels at all. When Ordered ACOL-PL is trained on nine out of ten CIFAR classes, still with ten clusters available, one should remain empty. When later (in the second stage) the tenth class is introduced to Ordered ACOL-PL, without subclass labels, it could be clustered into this empty spot, as this would allow the network to further optimise its Balance, and placing it in a cluster with a known class would increase the Affinity, which the network also tries to avoid. A potential caveat is that the network uses ReLU’s, which are prone to dying off when the weight becomes zero. As it first learns not to use this cluster, this could happen, and a leaky-ReLU should be used instead.

Ordered ACOL-PL could also be improved by looking at more recent literature. This work build on the ACOL layer proposed by Kilinc and Uysal, however, near the end of this work they revised their method. In this new version of their paper they have removed all mentions of the uninformative superclasses, and they refer to their GAR paper for the used loss function, where before they proposed this in the ACOL paper. This new loss does no longer use the coactivity term, and it is likely that the work done in this thesis could also benefit from this. For most experiments done here, the affinity value never dropped below the threshold and the coactivity therefore had no effect. In the
runs where this did happen, a problem with the thresholded coactivity term Kilinec and Uysal proposed in their original ACOL paper becomes clear. When the affinity drops below the set threshold, the coactivity term, which generally has a relatively large value at this point, is added to the loss function. The idea was that the network would then more strongly optimise the specialising property of the clustering nodes. Effectively, the network learns that whenever the affinity value drops below the threshold, the loss value increases, and therefore it learns to avoid this. Occasionally the network might overcome this, but generally this is not the case and this results in undesired behaviour. This is even already visible in the original paper by Kilinec and Uysal as shown in Figure 5.2, where the affinity clearly shows this fluctuating behaviour. Therefore removing the coactivity term in Ordered ACOL-PL might lead to an improvement, although the threshold only was reached consistently for the informative superclass settings on MNIST.

![Figure 5.2: Figure taken from Kilinec and Uysal showing that the affinity is fluctuating and evens out over time, well above the set threshold.](image)

Finally, Ordered ACOL-PL could be useful in many other ways. If a performance of Ordered ACOL-PL that is less dependent on the dataset or superclass choice is obtained, it could be used for a large variety of applications. One could for instance use it for object localisation, by determining the bounding boxes over the clustered subclasses rather than for the classes. A car from the side would have a different bounding box than from the front, and learning these different types of bounding boxes for a single object automatically could be of great use. Furthermore, by clustering the clustering of a superclass, a hierar-
chy could be discovered. For example: If the superclass label is ‘Animal’, one of the clusters ACOL could find is ‘Dog’, and this could then in turn be clustered into the different breeds of dogs. As discussed in the related work section (Section 2.4) such a hierarchy could be used to boost the accuracy in many different ways.
Ordered ACOL-PL can outperform or equal all baselines under any percentage of labelled data, providing a competitive performance with less labelling effort. This does depend on whether ACOL provides a good enough clustering by itself. The clustering formed by ACOL has a large influence and either increases or decreases the performance of Ordered ACOL-PL, depending on its individual performance. The usefulness of the clustering ACOL creates is strongly depended on the choice of superclasses, as it clusters on the feature space learned to correctly classify to the superclass. A superclass definition that directly relies on knowledge of the subclasses will lead to a more desirable clustering by ACOL than a superclass definition that can be learned trivially without knowledge of property the network should cluster on. No clear heuristic for the optimal superclass is found yet. Another factor on which the performance of Ordered ACOL-PL depends, is how strong a seed the first learning stage is able to learn. Overall, Ordered ACOL-PL makes efficient use of the feature space it is provided with, as it is capable of using a weighted feature space to cluster in.

Ordered ACOL-PL provides a promising path for semi-supervised image classification. If the correct conditions are met, or if the method is made more robust in future work, Ordered ACOL-PL can significantly reduce the labelling...
effort required to implement neural networks outside of academia. Computer vision is already being used to automate many tedious tasks, and reducing the time it takes to annotate the target data would speed up this process even more.


media@trecvid 2016 med and avs. In Proceedings of the 14th TRECVID
Workshop, volume 2016.

[28] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J.,
COCO: Common objects in context.

[29] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., &
Conference on Computer Vision (ECCV) (pp. 21–37).

based semisupervised learning. volume 100 (pp. 2624–2638).

Improving semi-supervised learning with auxiliary deep generative models.
In Workshop on Advances in Approximate Bayesian Inference (NIPS).


ral networks using natural pre-images. volume 120 (pp. 233–255).

[34] Marino, K., Salakhutdinov, R., & Gupta, A. (2016). The more you
know: Using knowledge graphs for image classification. arXiv preprint

[35] Markatopoulou, F., Mountzidou, A., Tzelepis, C., Avgerinakis, K.,
Gkalelis, N., Vrochidis, S., Mezaris, V., & Kompatsiaris, I. (2013). ITI-
CERTH participation to TRECVID 2013. In TRECVID Workshop,
Gaithersburg, MD, USA.

Reorganized pre-training for video event detection. In Proc. of the 2016
ACM International Conference on Multimedia (ACMMM) (pp. 175–182).: ACM.


