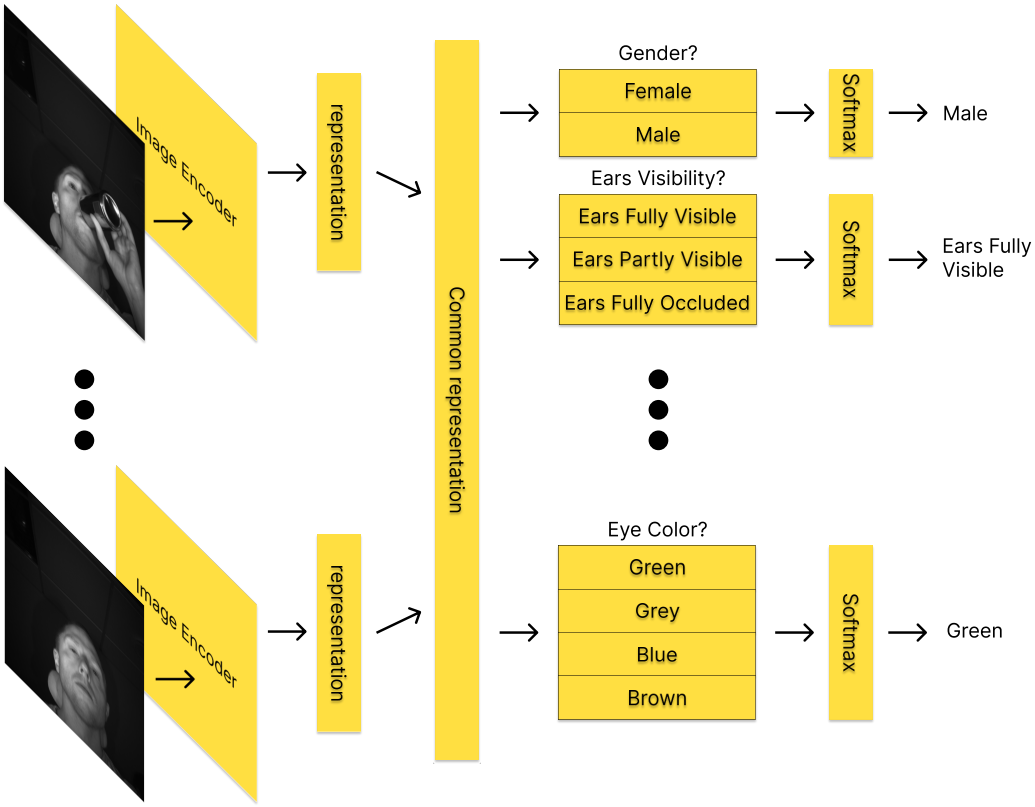
CHALMERS
UNIVERSITY OF TECHNOLOGY



MTL vs STL: NIR Video metadata classification using self-supervised semi-supervised learning

NIR Image multi-label classification

Master's thesis in Computer science and engineering

Adam Thörnblom
Gustav Fåhraeus

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

MTL vs STL: NIR Video metadata classification using self-supervised semi-supervised learning

NIR Image multi-label classification

Adam Thörnblom
Gustav Fåhraeus

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

MTL vs STL: NIR Video metadata classification using self-supervised semi-supervised learning
NIR Image multi-label classification
Adam Thörnblom
Gustav Fåhraeus

Advisor: Joel Lindkvist, Smart Eye
Examiner: Lars Hammarstrand, Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An image visualizing the architecture of the proposed model's network.

Typeset in LaTeX
Gothenburg, Sweden 2023

MTL vs STL: NIR Video metadata classification using self-supervised semi-supervised learning
NIR Image multi-label classification
Adam Thörnblom
Gustav Fåhraeus
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The combination of self-supervision and semi-supervision has emerged as a popular research topic in recent years. However, existing studies primarily focus on single-task models trained on datasets where individual images are labeled with a single class, overlooking the challenges associated with multi-class scenarios. In this thesis, we propose a modified S4L framework, which is a self-supervised semi-supervised learning framework specifically designed to handle partially labeled data. The modified S4L framework addresses some limitations of previous approaches and demonstrates its effectiveness in both multi-task learning and single-task learning settings. The research focuses on the classification of visual attributes in human subjects, specifically in near-infrared (NIR) images.

Keywords: Self-supervised learning, Semi-supervised learning, Multi-task learning, Single-task learning, NIR Video Classification.

Acknowledgements

We express our sincere gratitude to Smart Eye for their collaboration in conducting this thesis. Special thanks goes to Joel Lindkvist, our supervisor at Smart Eye, and the rest of the AS-core team for assistance throughout the thesis. We would also like to extend our gratitude to our Examiner, Lars Hammarstrand, for his guidance and support.

Adam Thörnblom, Gustav Fåhraeus, Gothenburg, 2023-06-19

Contents

List of Figures

List of Tables

1

Introduction

Managing extensive and evolving datasets is not a trivial task. As time progresses organizations collect more data and the usage of the data also evolves over time. Additionally, there might be different parties utilizing and contributing to the same dataset, which can lead to an increased dataset complexity if the parties' annotation processes differ. How can this process of managing datasets be made easier?

Working with data is in general a difficult, expensive, and time-consuming task. Current research estimates that over 80% of engineering tasks in machine learning (ML) projects concern data preparation and labeling [1]. To alleviate this time-sink, organizations often use either third-party labeling systems or in-house labeling systems, both with their own benefits and issues. This thesis deals with a complementary approach which minimizes the need for human supervision: data labelling using ML, more specifically using self-supervised/semi-supervised learning. Self-supervised learning refers to when pretext tasks are defined that a machine learning model trains on to get better on the actual task we are interested in. Semi-supervised learning refers to when the training incorporates labeled as well as unlabeled examples. For a more extensive description about self-supervised semi-supervised learning, refer to Section 2. Specifically, metadata about video recordings of people made for the purpose of training AI systems at the company Smart Eye is annotated. In Figure 1.1 example images are shown from a sample of recordings from this dataset. Metadata refers to "data about data", and is used to classify, organize, label, and understand data, ultimately making sorting and searching for specific data much easier.

Formally in this thesis, a self-supervised semi-supervised learning framework is developed for the task of metadata annotation. In addition an investigation is made regarding whether multi-task self-supervised semi-supervised learning outperforms single-task self-supervised semi-supervised learning in the specific context of near-infrared (NIR) data recorded for the purposes of driver monitoring. Essentially the comparison concerns whether training a single machine learning model to do multiple things (multi-task) outperforms training multiple individual models to perform individual tasks (single-task). Specifically, the aim is investigate whether training a single multi-task model to predict all new labels simultaneously is more effective than using multiple single-tasks models to predict each individual label.

(a) In Car (b) In Lab (c) In Lab

Figure 1.1: Three example NIR images from the dataset. The recordings are either done in cars or in a lab environment.

Ultimately, a multi-image multi-task model trained using a modified Google Brain S4L framework is proposed to be applied in Smart Eye's specific context for the problem of metadata labeling using ML [2].

1.1 Problem Description and Collaboration

This thesis is done in collaboration with Smart Eye. Smart Eye is a Swedish technology company that specializes in AI eye-tracking solutions and driver monitoring systems. These systems are trained using their large, extensive, and continuously growing datasets. The company consistently records new video data to improve and tweak the performance of their systems. These systems, such as advanced driver-assistance systems (ADAS), rely heavily on the fact that the underlying machine learning models are robust over a variety of different inputs, i.e model bias is minimized [3]. This is only ensured if the model is trained on a diverse dataset, which means that metadata (environment factors and subject attributes) is an important aspect to analyze model behavior for different types of inputs. However the metadata annotation at Smart Eye has not been consistent through-out the company's existence. The consequence of this is that the majority of the recordings have partially labeled metadata, i.e there are a lot of missing labels.

Essentially the ideal state is the one where all labels are present for all data entries. To accomplish this the fact that some metadata labels refer to visual elements could be used. For example consider in Figure 1.2 that the labels are referring to different clothing items that a person in an image is wearing. Then some images could have the "Wearing Hat" label set (indicating that the person is wearing a hat), while others have the "Wearing Glasses" label set (indicating that the person is wearing glasses). The ideal state is the one where all images have both the Wearing Hat and Wearing Glasses label set, and this would apply for all labels relating to visual features present in the metadata.

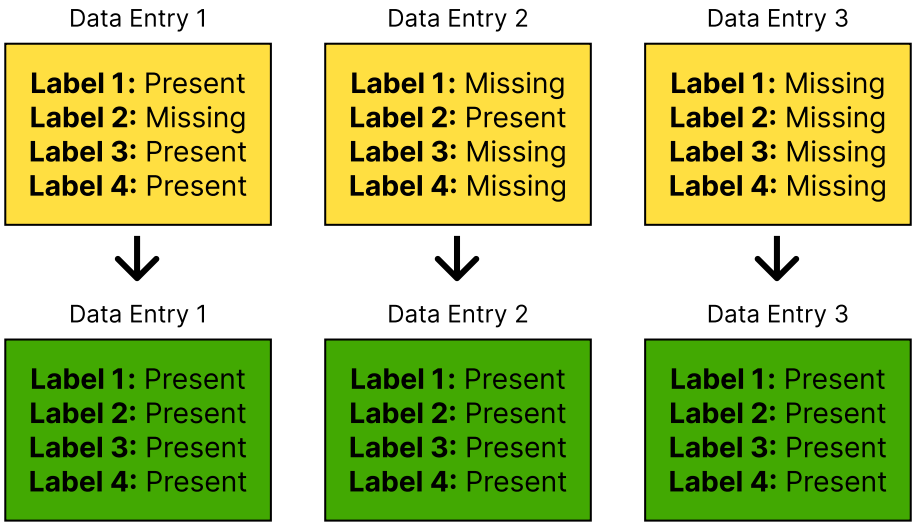


Figure 1.2: An illustration of the missing labels problem. In the first layer an original example dataset can be seen and in the second layer the ideal state is seen, i.e where all entries have all labels.

To summarize, we want to efficiently utilize the complete dataset to train a model that can accurately annotate the missing metadata labels. Additionally, we want to compare whether multi-task or single-task learning is the better approach for this problem.

1.2 Overview of Approach

A possible solution to the problem in the section above , assuming some labeled data is accessible, is to use a supervised model for training. However, this approach falters if there are few labeled examples available. The reason for this is that ML models do not perform well unless trained on diverse data, and the data is unlikely to be diverse enough when working with a small amount of examples [3]. Finding ways to better leverage unlabeled data, i.e semi-supervised learning, has therefore been a hot research topic in the last decade to mitigate this problem [4]. The approach being dealt with in this thesis is to use what is called self-supervised semi-supervised learning, which is a combination of the two machine learning techniques self-supervised learning and semi-supervised learning. These two techniques alleviate the problem of having a few labeled examples by leveraging both labeled and unlabeled dataset entries during training.

Furthermore, since the problem being dealt with concerns multiple different labels, multi-task learning (MTL) is another ML technique that could be applied. When doing MTL, a single model is trained to perform multiple tasks simultaneously instead of training separate models for each individual task (i.e each label that should be predicted in this case). The idea is that when doing multi-task learning the shared features and relationships between tasks is leveraged to improve overall performance. Training a more general model has been shown in other contexts to

lead to improvements in classification performance [5], and if this applies to this specific context is part of what is being investigated in this thesis.

2

Theory

The purpose of this chapter is to provide an overview of the theoretical concepts that the research presented in this thesis builds upon.

2.1 Neural Networks

Neural networks (NNs), more formally called artificial neural networks, are computing systems where the solutions to a problem is learned from a set of examples. A neural network can be regarded as a non-linear mathematical function that maps a set of input variables to a set of output variables. This mapping depends on a set of parameters, θ, whose values is determined on the basis of a set of given examples. This process of determining the values of θ is called *learning* or *training*. Once the weights have been fixed (i.e the training is done), then new data can be processed by the network very quickly [6].

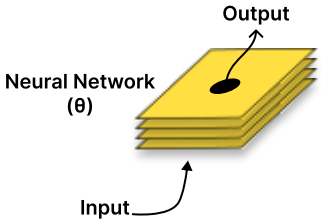


Figure 2.1: An illustration for a model of a neural network.

In addition to their high processing speed, neural networks have the capability of learning a general solution to a problem from a set of specific examples. This circumvents the need to design and develop a specific solution which might be difficult, or even impossible [6].

The principal disadvantage of using neural networks is that they by construction require data. Additionally, if a model is to be useful then this data needs to be representative of the intended inputs of the final model. For example, consider a model trained to identify in which images cats appear, but the model is only exposed to images where black cats appear. If the model was to be tested on two different datasets, one set where only black cats appear, and one set where appearing cats

follow the actual distribution of different cats, then the performance of the model would be different. In short, neural networks are a product of the data they are trained on.

Assuming that the training data distribution and actual data distributions are similar, then neural networks are excellent at solving problems that have the following characteristics:

1. There is a lot of data for network training.
2. There are patterns in the data that can be exploited for learning.
3. It is difficult to construct a simple first-principle solution which is adequate.
4. New data must be processed at high speed.
5. The data processing method needs to be relatively robust to input noise.

The networks themselves are mainly compromised of a high number of interconnected computational nodes, often referred to as neurons. The structure of a basic network is shown in Figure 2.2. Neurons are grouped together in to layers which are either the input layer, the output layer, or some hidden layer. The input itself is usually of the form of a multidimensional vector which is distributed to the hidden layers. The hidden layers will then make decisions from the previous layer and weigh up how a stochastic change within itself detriments or improves the final output, this is the process of learning. If there are more than one hidden layer stacked upon each-other then the process is called deep learning.

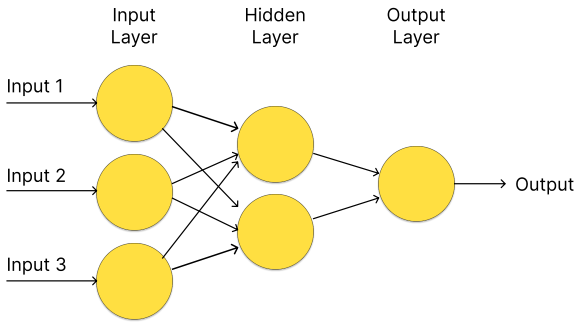


Figure 2.2: A simple three layered feedforward neural network, comprised of a input layer, a hidden layer, and an output layer.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of NNs that are often used in the field of pattern recognition within images. The architectures of CNNs are designed with image specific-features built into the network, making them more suited for image-focused tasks.

One limitation of the traditional feedforward networks is that the models get too complex too quickly in relation to the inputs they receive. For example consider the commonly used MNIST dataset of handwritten digits, from which an example

is given in Figure 2.3. Entries in this dataset have a relatively small size of just 28 x 28 pixels. However with this dataset in mind a single neuron in the first hidden layer of a traditional NN, such as the one appearing in Figure 2.2 would require 784 weights (28 x 28 x 1), assuming the values have been normalised to just black and white values. If a slightly larger image input of (64 x 64) is considered, the number of weights would increase substantially to 12 288.

Figure 2.3: The digit 4 such as it appears in one image of the MNIST dataset, a popular and basic benchmarking dataset for image classification.

Generally feedforward networks grow too large as the inputs themselves grow larger, the network needs to be larger to deal with the scale of the input [7]. This becomes a problem because the networks eventually require too much computational power and time to train. Furthermore, the bigger and more complex the network becomes, the more prone it is to overfitting, i.e the network being unable to learn and generalize effectively. CNNs essentially deal with this by their architecture being set up in a way to counteract the large inputs.

CNNs are comprised of three different layers (aside from the input/output layers): convolutional layers, pooling layers, and fully-connected layers.

The convolutional layer's goal is to extract the relevant features from input data that has a grid-like structure, such as images or time series data. The layer performs an operation known as convolution across the images. They do this by applying a set of filters, also called kernels, to the input. The filter performs a local, element-wise multiplication between its weight and the local region of the data, and then proceeds to slide over the entire input to compute a new set of values. This results in a feature map, which highlights different patterns in the data at different spatial locations.

This concept of using a filter can be extended to where multiple filters are used in parallel to detect features across the inputs. These features can be as simple as edges or corners in the initial layers, and then get increasingly complex as the data is processed in the network.

The convolutional layers' purpose is to capture meaningful spatial information and to enable the network to learn meaningful representations to facilitate the extraction of important features for tasks like image classification and object detection.

The pooling layer on the other hand is used is to reduce the spatial dimensions (width and height) of the input volume, while retaining the integrity of the information present in the data. Pooling operates on each feature map independently, using the same idea of a filter that moves across the input. The most commonly used type of pooling is max pooling, which selects the maximum value within each window and discards the other values. Some other types of pooling include average pooling and and sum pooling.

Finally the fully-connected layers will then perform the same duties found in standard NNs and attempt to produce class scores from the activations to be used for classification or other tasks.

2.2 Machine Learning and Optimization

Machine learning is a technique that improves system performance by learning from experience via computational methods. In computer systems, experience comes in the form of data, and the main task of machine learning is to develop learning algorithms that build models from data. By feeding the learning algorithm with data, a model is obtained that can make predictions [8].

Optimization refers to process where a model is trained iteratively which results in a maximum and minimum function evaluation. The optimization process involves finding the best set of parameters or weights for the model that minimizes the error, or loss, between the predicted output and actual output.

2.2.1 Regularization Techniques

When training neural networks there are two problems that one has to deal with that live on opposite ends of a spectrum: underfitting and overfitting. Underfitting refers to when poor model design and inefficient optimization occurs and thus the predictions of the model are prone to large errors. On the other end is overfitting, where the model learns the training data too well or captures noise and irrelevant patterns in the data, which can lead to a big discrepancy between the training error and the test error. Regularization refers to the process where additional information is introduced in to the model to minimize this discrepancy [9]. Basically regularization is applied in an attempt to improve generalization and thus improve model performance.

Some common regularization techniques are L1 and L2 regularization, which encourages the model's weights to be smaller and thus preventing over-reliance on specific features [10]. Furthermore, another example would be to incorporate neuron dropout in to the model, which randomly deactivates a fraction of the neurons in the network during each training iteration [11]. This is similar to L1 and L2 regularization in that it tries to prevent over-reliance on specific parts of the network. Different loss functions can also be used as regularization to encourage specific model behavior. These are just a few examples, the choice of which regularization technique to apply generally depends on the specific problem, model architecture, and available data.

2.2.2 Loss Functions

In deep learning, a loss function is used to quantify the difference between the predicted output and the true output for a given input. Basically it is a method of evaluating how well the model models the dataset. If the predictions of the model are inaccurate, the loss function outputs a higher number, if they are accurate, it outputs a lower number. Typically the loss function is minimized, i.e a smaller loss

equates better model accuracy. A simple example of a loss function is the Mean Square Error (MSE). To calculate the MSE, the difference between the model's predictions and the ground truth is calculated, squared, and then averaged across the whole dataset. If the model that used this loss function had only completely accurate predictions, then the loss would be 0.

The goal of training a neural network is to find the optimal set of model parameters that minimizes the loss function, also called learning objective, over a training dataset. Different types of loss functions can be used depending on the problem at hand. In this section, the loss functions that are used in this thesis will be described.

2.2.2.1 Categorical Cross-Entropy (CCE)

The Categorical Cross-Entropy (CCE), also called Softmax loss, is a common loss function for multi-class classification problems. It measures the difference between the predicted probability distribution and the true probability distribution over the classes. It is applicable whenever the model has a Softmax activation layer and Cross-Entropy loss is applied. The Softmax layer outputs class probabilities which is essential for the calculation [12]:

$$\text{CE} = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{2.1}$$

where e^{s_p} is the score for the positive class and e^{s_j} refers are the scores inferred for each class in C.

The CCE loss function is defined as the negative log-likelihood of the true class labels given the model's predicted probabilities for each class. By minimizing the negative log-likelihood, the model is encouraged to assign high probabilities to the correct class and low probabilities to the incorrect classes. The CCE loss function is widely used in deep learning for image classification, object detection, and natural language processing tasks, among others [3].

2.2.2.2 Virtual Adversarial Training (VAT)

Virtual Adversarial Training is a regularization technique based on virtual adversarial loss [9]. The loss is defined as the robustness of the conditional label distribution around each input data against local perturbation. The goal is to encourage that entries that are close to each other in the input space should also be close to each other in the output space. It does so by introducing a new loss term to the objective function, which encourages the model's output to be invariant to small adversarial perturbations of the input [9].

This method is based on Adversarial Training (AT) [13], however AT requires full label information to be used while VAT does not. This allows VAT to be applicable in a semi-supervised learning context, which is why it has been chosen rather than AT in this thesis. Concretely, the VAT loss for a model f_θ is:

$$L_{\text{VAT}} = \frac{1}{D_u} \sum_{x \in D_u} \text{KL}(f_\theta(x) \,||\, f_\theta(x + \Delta x)), \tag{2.2}$$

where

$$\Delta x = \arg\max_{\delta} \text{KL}(f_\theta(x) \,||\, f_\theta(x + \delta)).$$

The KL-function refers to Kullback-Leibler divergence (also called relative entropy) [14]. KL divergence is denoted $KL(P \,||\, Q)$ and is a measure of the relative entropy in information in two distributions, or in other words how different they are. A simple interpretation of the KL divergence of P from Q is the expected excess surprise from using Q as the model when the actual distribution is P. A KL divergence of 0 indicates that the two distributions have identical qualities of information. D_u in (2.2) refers to the complete set of labeled and unlabeled images in the dataset.

The KL divergence is calculated as:

$$KL(p(x) \,||\, q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx \tag{2.3}$$

The calculation of Δx is supposed to be done to find the δ which maximises KL-divergence. In reality this is never done since it would cost too much computational resources, therefore an approximation is done which instead comes at the cost of an extra forward and backwards pass for every optimization step [2]. The way in which δ is explored can be limited by constraining it $|\delta| < \epsilon$, where ϵ is a hyperparameter.

2.2.2.3 Conditional Entropy Minimization (EntMin)

Conditional Entropy Minimization is another regularization technique which allows for the incorporation of unlabeled data during the training process. It adds a loss to the objective function which, when minimized, encourages the model to make confident predictions on unlabeled data. Concretely the conditional entropy minimization is calculated as [2]:

$$L_{\text{EntMin}} = \frac{1}{|D_u|} \sum_{x \in D_u} \sum_{y \in Y} -f_\theta(y|x) \log f_\theta(y|x) \tag{2.4}$$

One fundamental assumption when using the EntMin on unlabeled data is that it indeed has the labels that the model is trained on, even when the particular classes are not known during training.

EntMin is seldom used as a stand-alone loss in the context of neural networks because the model can circumvent the loss by increasing the weights of the last layer. Essentially this means that the model will not explicitly enforce class separation or discriminate decision boundaries. One way to resolve this is to incorporate a goal in the learning objective that the model's predictions should be locally-Lipschitz, which means that the outputs of the model exhibits a certain level of stability and boundedness in the vicinity of each input point (essentially that the inputs are not sensitive to small perturbations). Since VAT does exactly this the two loss functions are often considered together [15].

2.2.3 Optimization Algorithms

The goal of optimization is finding the best (or a suitable alternative) solution among many feasible solutions that are available to us. A feasible solution is a solution that satisfies all the defined constraints of the optimization problem. In the context of training a neural network optimization refers to the process of finding the best set of parameters for the network that minimizes the loss functions.

There are multiple different approaches when it comes to optimization. In this section the classic stochastic gradient descent method is described, as well as the optimization algorithm that was used during the thesis, ADAM.

2.2.3.1 Stochastic Gradient Descent (SGD)

The goal of gradient descent is to descend a slope to reach the lowest point on the surface. In Figure 2.4 a simple example objective function is shown to illustrate how SGD works, in reality there are often a lot more variables to consider than just X and Y.

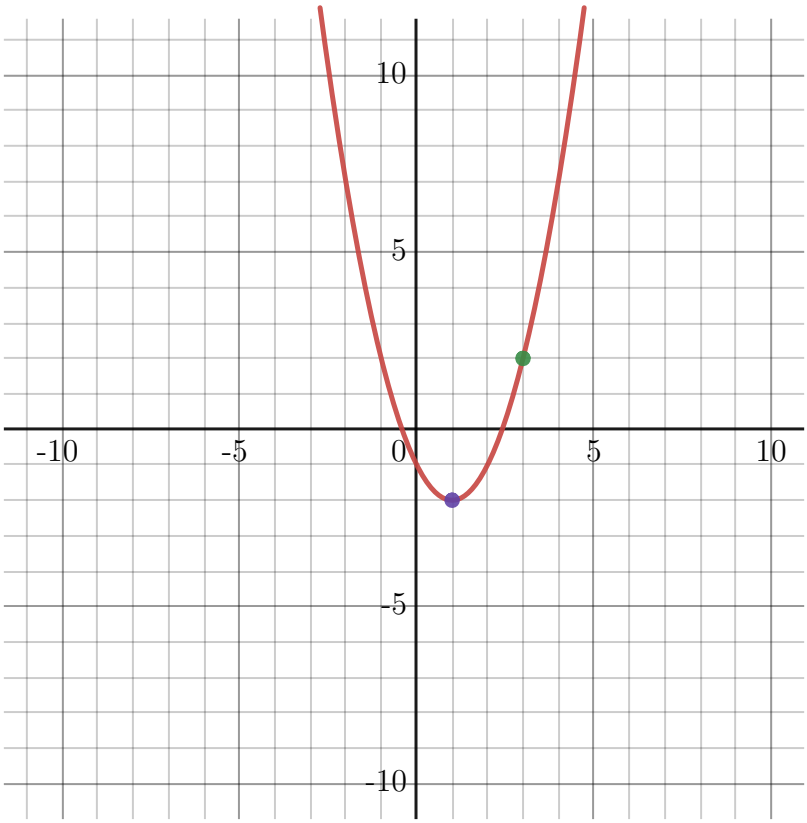


Figure 2.4: The function $y = x^2 - 2x - 3$. The lowest point on the graph is (1,-2)

SGD starts by randomly initializing the model's parameters, by doing this the model will end up in some arbitrary point on the graph, for example the green dot (3,2). To move down the line the algorithm finds the slope of the objective function with respect to each feature, i.e the gradient of the function is computed. This gradient

then indicates which direction to move to go down the surface. The size of the step is a hyperparameter that can be tuned which will affect how quickly the algorithm converges. This process is repeated until the lowest point is found.

What has been described so far is just a general gradient descent algorithm. The downside of this approach is that when the number of data points and features increases, this calculation of the gradient becomes very expensive. To alleviate the need for going through all data points, SGD instead randomly picks one data point and computes the gradient for this point. There are also methods that incorporate a small set of data points instead and this would be refered to as "mini-batch" gradient descent.

2.2.3.2 Adaptive Moment Estimation (ADAM)

The ADAM optimizer is common optimization algorithm that can be an alternative for the stochastic gradient descent process. The name is derived from adaptive moment estimation. The benefits of using ADAM over regular SGD is that ADAM dynamically adjusts the learning rate for each feature during training, SGD does not do this which typically means that a fixed learning rate is used that might need manual tuning. ADAM also incorporates momentum which can help accelerate and overcome local minima. The way it does this is by keeping track of previous gradients and use them to influence the current update. Generally ADAM tends to converge faster, while SGD often converges to more optimal solutions [16].

2.3 Multi-task Learning

In multi-task learning (MTL) a single model is trained to perform multiple tasks simultaneously. MTL aims to leverage useful information contained in multiple learning tasks by making them share a common feature encoding, by doing this the idea is that it is possible to exploit commonalities and differences across tasks with the intention of improving overall model performance. Multi-task models also require less time training and less maintenance since only one model is being dealt with rather than multiple models.

To get an intuition for why MTL works, imagine students studying for multiple exams that share some common topic. If a person studies for each exam separately they might miss out on connections between the topics and waste time learning the same information twice. By studying for them all at once the person can identify shared topics and focus on learning them effectively, which may save time and improve the overall performance on all exams.

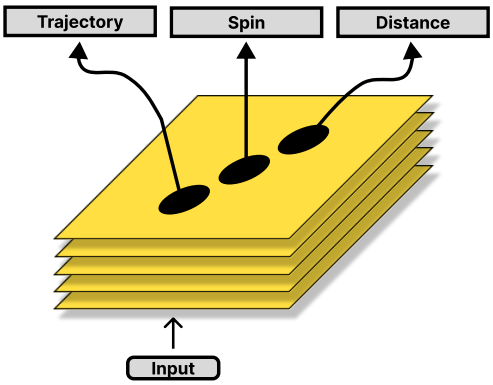


Figure 2.5: An illustration of a multi-task model receiving the same input as the single-task models in Figure 2.6 and outputting all three different metrics.

Furthermore, in another context such as golf, imagine a model trained to predict features concerning the flight of a golf ball that has been struck. Using an MTL approach to predict the trajectory, spin, and distance of a swing (such as in Figure 2.5) is reasonable because these three factors are interconnected and impact one another. For example, a change in the spin of the ball can affect its trajectory and distance. If a single-task approach was used, such as the one in Figure 2.6 then the model might miss out on relationships between the factors and produce less accurate predictions.

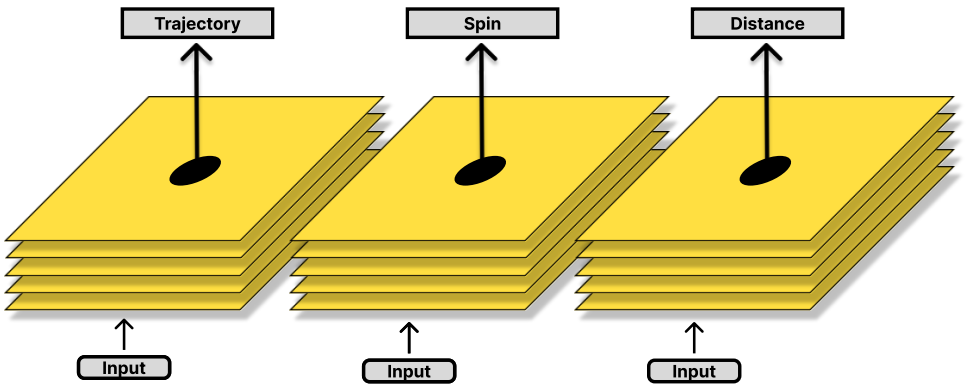


Figure 2.6: An illustration of three single-task models receiving the same input and outputting three different metrics.

Under the assumption that at least a subset of the tasks are related, jointly learning multiple tasks is empirically and theoretically found to lead to better performance than learning them independently [17]. MTL can be applied in several ML

paradigms, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, etc.

2.4 Self-supervised Learning

Self-supervised learning is part of the unsupervised learning paradigm, where the objective of the ML models is to learn from data without relying on explicit supervision in the form of labels or other forms of annotation. Specifically self-supervised learning techniques define pretext tasks which can be formulated using solely unlabeled data, but still requires the model to achieve a higher-level semantic understanding in order to be solved. These pretext tasks can take on many forms, such as predicting the degree of rotation applied to an image, detecting whether it has been horizontally flipped, or identifying the area of an image that has been cropped out (or really any transformation that can clearly be defined). By training on such pretext tasks, the model can learn to capture the underlying patterns and relationships in the data, which can be leveraged for downstream tasks.

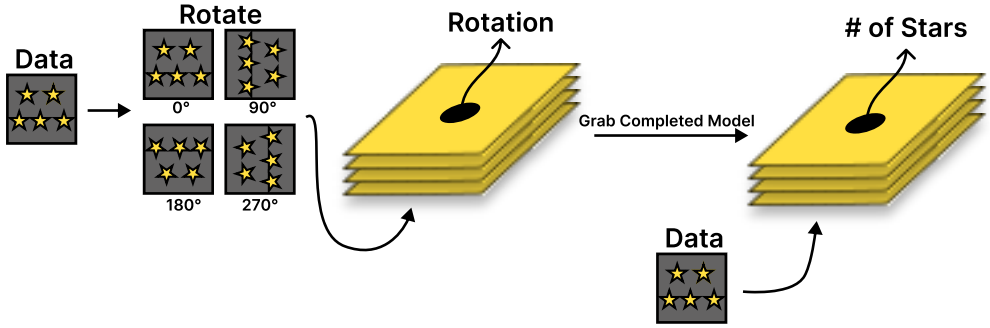


Figure 2.7: An example of what a self-supervision pipeline could look like.

In Figure 2.7 an example is made of how self-supervision learning can be applied. Every image in a dataset has been rotated by 90° until four different versions of each image have been created with a different rotation applied (0°, 90°, 180°, 270°). A model is then trained on this extended dataset to predict which rotation has been applied. This model is then used for a different task which is to predict the amount of stars that is present in the image. The idea is that from learning to identify which rotation has been applied, the model will also get better at identifying how many stars are present in the images. This specific technique of rotation is used in this thesis.

Self-supervised learning has shown remarkable success in achieving state-of-the-art results in various domains and has become an increasingly popular research topic in recent years [18]. However despite demonstrating encouraging results, purely self-supervised techniques learn visual representations that are significantly inferior to

those delivered by fully-supervised techniques. This is why self-supervision is often combined with semi-supervision, where the model is provided with a few labeled examples in addition to the unlabeled data [2].

2.5 Semi-supervised Learning

In many scenarios where ML scenarios there is not a shortage of data, but rather a shortage of labeled data. In these scenarios a semi-supervised learning approach can be applied, which conceptually is an approach situated between supervised and unsupervised learning. Semi-supervised learning combines supervised learning and unsupervised learning with the intention to improve the performance of one of these two tasks by utilizing information generally associated with the other [19]. For example, consider a classification problem where additional unlabeled data points is used to aid in the classification process, or consider a clustering problem where the learning procedure might benefit from the knowledge that certain data points belong to the same class.

In semi-supervised classification scenarios where the goal is to increase the amount of labeled data there are two main ideas: consistency regularization and pseudo-labeling. In consistency regularization an auxiliary objective task is added to the ML model, and this task is added to encourage the model to learn properties of the input that are related to the primary task. Additionally self-supervision, which is discussed in Section 2.4, is a form of semi-supervision learning since it enables a model to learn from both labeled and unlabeled data. Pseudo-labeling on the other hand, refers to the idea of assigning labels to unlabeled data points by using the models predictions on those data points, which is then used during the supervised training.

2.5.1 S4L

S4L [2], or Self-Supervised Semi-Supervised Learning, is a paper published in 2019 that tackles the problem of semi-supervised learning of image classifiers. The S4L framework was presented in a paper published by the Google Brain team in 2019. The team members mention in the paper that the key take-away from the paper is that semi-supervised methods can greatly benefit from being combined with self-supervised methods in regards to model performance.

In S4L the complexity surrounding incomplete datasets is discussed. To help alleviate this problem the authors suggest a learning approach that can successfully learn to recognize new concepts by leveraging only a small amount of labeled examples. Additionally, to still be able to leverage the vast amounts of unlabeled data that often is available they incorporate self-supervision in to the model. This combined approach is summarized as a learning objective of the following form:

$$\min_{\theta} L_l(D_l, \theta) + wL_u(D_u, \theta), \tag{2.5}$$

where L_l refers to a standard cross-entropy classification loss which is calculated for all labeled images in the available data, and L_u refers to an unsupervised loss that is applied to different instances during different phases of the training, additionally a rotation CCE loss (or some other self-supervised transformation loss) is calculated for every image at different stages. The unsupervised loss is L_u is the sum of a Virtual Adversarial Training (VAT) loss L_{vat} and a Conditional Entropy Minimization (EntMin) loss L_{entmin}. In Section 2.2.2 these loss functions are explained more in depth. This means that L_u can be written as:

$$L_u = w_{\text{vat}} L_{\text{vat}} + w_{\text{entmin}} L_{\text{entmin}} \tag{2.6}$$

Where w represents non-negative scalar weights and θ refers to the parameters of the model being trained. The learning objective (2.5) is optimized using some variant of stochastic gradient descent that uses mini-batches to update the parameters θ. These mini batches take on the form of $x_l, y_l \in D_l$ and $xu \in D_u$ and can take technically take on on any size, although in the authors' experiments they use mini-batches of equal sizes.

The authors try a variety of approaches, but the top performing model is the one they call the *Mix Of All Models* (MOAM), and the training of this approach follows 3 steps: in the initial Step 1 the learning objective (2.5) is optimized. Then this trained model is used in order to generate pseudo labels for the unlabeled images for Step 2 which is done in the exact same fashion as Step 1. Ultimately, in Step 3 the model is fine-tuned using only the true labels.

While training on the semi-supervised dataset ILSVRC-2012[20] with 10% of the labels S4L scored 91.23% Top-5 accuracy and 73.21% Top-1 accuracy, which was a new state-of-the-art result at the time. Top-5 accuracy considers a classification correct if any of the five top predictions matches the target label.

2.6 Related Work

Self-supervision has been successfully applied together with semi-supervision before in different contexts [2], [21]–[25]. In this thesis the S4L framework [2] is built upon since it can easily be modified to fit the context of the thesis.

3

Methods

This chapter provides a comprehensive overview of the approaches and techniques used in this thesis. This chapter is essential for understanding the reasoning behind the implementation.

3.1 Data

Smart Eye provided a subset of their data, in the form of recordings in their own proprietary format. Each recording can consist of multiple views from different cameras simultaneously. This means that for each timestamp of the recording one frame can be provided from each of the available cameras. For our thesis we will sample images in .pgm/.png format and metadata in .json/.csv format from these recordings. The level of metadata annotation is not consistent across these recordings, some entries contain zero labels, some a subset, and some are fully annotated. The recordings are recorded in near-infrared (NIR), which is a spectrum invisible for humans. The reason to record in this spectrum is that it allows for the use of IR-flashes which do not disturb the driver and reduces the influence of other light sources such as daylight, car lightning etc.

3.1.1 Metadata Labels

The metadata labels used in this thesis have multiple classes, which are used to categorize and classify data based on its attributes. This means that the model(s) will predict the probabilities for each class within the label. Specifically, the labels and their classes are:

- **Gender:** "Female", "Male"
- **Car Recording:** "In Car", "In lab"
- **Ears Visibility:** "Ears Fully Occluded", "Ears Partly Visible", "Ears Fully Visible"
- **Chin Visibility:** "Chin Fully Visible", "Chin Fully Occluded"
- **Glasses Kind:** "Glasses Kind None", "Glasses Kind Transparent", Glasses Kind Semi-Transparent", "Glasses Kind Blocking".
- **Mouth Visibility:** "Mouth Fully Visible", "Mouth Fully Occluded"

- **Nose Visibility:** "Nose Fully Visible", "Nose Fully Occluded"
- **Eye Color:** "Green", "Blue", "Brown", "Gray"
- **Face Mask Kind:** "Face Mask Kind None", "Face Mask Kind Black", "Face Mask Kind White", "Face Mask Kind RGB"
- **Wearing Glasses:** "Wears Glasses", "Wears No Glasses"
- **Wearing Face Mask:** "Wears Face Mask", "Wears No Face Mask"

The labels "Wearing Face Mask" and "Wearing Glasses" were not included in the metadata originally, they were created by label propagation specifically for thesis based on whether a person was wearing a face mask or glasses, respectively. This was done because it should be easier to predict whether someone is wearing glasses or a face mask, rather than specifying the color or type of face mask or glasses. The most important factor (in the case of Smart Eye) is determining whether a person is wearing a face mask or not, rather than the specific type of face mask. Therefore, the output of the "Face Mask Kind" and "Glasses Kind" labels is only relevant if the "Wearing Face Mask" or "Wearing Glasses" label is true.

Furthermore there were additional cases where the relationships between the labels was used to classify the data. To accomplish this, label propagation was used. For instance, it was observed that wearing a face mask results in the full occlusion of the mouth and chin. Therefore, this relationship was utilized to propagate the label information across different attributes were label information was missing.

3.1.2 Datasets

While working with a balanced dataset is ideal, it is not always feasible, and often does not reflect the real-world scenario. The provided data suffers from both label and class imbalances. To prevent the model from being biased towards any particular label or class, a weighting scheme is employed. This ensures that each class and label receives a weight proportional to its inverse frequency, thereby giving more weight to the minority classes and labels and balancing the impact of each class and label on the model's training and evaluation process. Basically the model is rewarded more for predicting the minority classes correctly.

In this thesis, two datasets were constructed from the provided recordings to explore the effectiveness for different model inputs, Dataset_v1 (see Section 3.1.2.2) and Dataset_v2 (see Section 3.1.2.3). Each dataset was created in a way that ensures that a recording subject is present in only one set of the Training/Validation/Test sets. By enforcing this constraint, the model was not exposed to the subjects it was being evaluated on during the training. This method was adopted to ensure that the results are unbiased and generalize well to unseen data. Both datasets have the same split of recording subjects.

A 70/10/20 Training/Validation/Test split was opted for in this thesis, with the test set being larger than the validation set to ensure that the test score is robust and representative of the model's true performance, in [2] it is shown that a tiny

validation set in sufficient. However, achieving a balanced test set was not possible due to the original distribution of labels and classes in the provided recordings from Smart Eye. Thus, we decided to maintain the original distribution of labels and classes in each dataset as closely as possible. Another constraint on the sets is that we also want the sets to have their proportional share of recordings, and their proportional share of recording subjects, i.e the training set should contain 70% of the recordings and 70% of the recording subjects etc.

3.1.2.1 Solving the Train/Val/Test Split

The problem of selecting a subset of the recording subjects such that the subsets make up their proportional share of the recording subjects and recordings can be defined as follows: given a dataset of recording subjects, where each subject has a different number of recordings, we want to select a subset of recording subjects (and thus corresponding recordings) such that the subset includes approximately $z\%$ of all recording subjects and $z\%$ of all recordings. If no exact solution exists, we allow a deviation of up to 2% in the number of selected recording subjects and up to 3% in the number of selected recordings. These numbers were chosen such that the split we end up with is as close to possible to the ideal case, while allowing for a small margin of error to make the split easier to find.

This problem is similar to the Subset Sum problem, where the task is to find whether there exists a subset of a given set of numbers whose sum equals a specific target value [26]. Similarly, in our problem, we need to select a subset of recording subjects and their corresponding recordings while ensuring a specific distribution of classes/labels while maintaining the original distribution of classes/labels distribution. Since it is possible to reduce our problem of selecting recording subjects such that the sets make up their share of recording subjects and recordings from the Subset Sum problem, this makes the problem NP-complete [27]. Therefore when adding the additional requirement to keep the original distribution, this is also NP-complete.

Although finding an optimal solution which may not even exist for the problem can be both computationally expensive and time-consuming, it can be formulated as a linear programming (LP) problem which can be done in reasonable time [28].

To formulate the LP problem we start by defining a binary vector x of length n, where n is the total number of recording subjects available. The total number of recordings is defined as m, each recording subject is encoded as a integer vector v_i of fixed length k representing the class annotations for the recordings for that recording subject. We can stack all the v_i vectors into a matrix V, where each row corresponds to a recording subject. Then, we can obtain the original distribution of classes O by summing up all the rows of V.

To achieve a desired percentage z of each class in the selected recording subjects, we can multiply O by z to obtain the target distribution T. The LP problem aims to find a binary vector x that minimizes the absolute difference between the weighted sum of the rows of V with x as coefficients, and T.

Formally, we can formulate the LP problem as follows:

$$\min \sum_{i=0}^{k} |T_i - \sum_{j=0}^{n} x_j V_{j,i}| \tag{3.1}$$

Subject to the following constraints:

$$\sum_{x} \leq n(z+2)$$

$$\sum_{x} \geq n(z-2)$$

$$x \cdot M \leq m(z+3)$$

$$x \cdot M \geq m(z-3)$$

Where k represents the total number of classes, and M is a integer vector that represents the number of recordings available for each recording subject. The different constraints ensure that the total number of selected recording subjects is within a specific range, while the minimization objective aims to minimize the difference between the weighted sum of the selected recordings and the target distribution.

In other words, the LP problem seeks to select a subset of recording subjects that collectively match a desired class distribution while satisfying certain constraints on the number of selected recording subjects. The objective is to minimize the discrepancy between the selected recordings and the target distribution. The vector M provides information on the number of recordings available for each recording subject, which can be used to ensure that the selected subset of recording subjects does not exceed the available recordings for each subject.

3.1.2.2 Dataset_v1

For each recording, the recording was divided into as many bins as possible, each bin containing 600 frames. Then depending on the recording's length a number of bins was discarded from the beginning of the recording and at the end to avoid sampling frames that does not contain the subject or frames that contain initial setup calibration. From each bin a frame was then sampled at random and was given its recordings metadata labels.

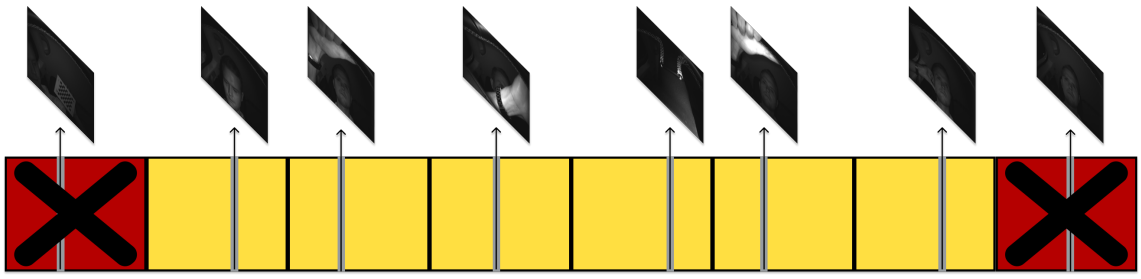


Figure 3.1: Illustration of the frame sampling process for Dataset_v1, visualizing the frame sampling problem as well as discarding the first and last bin. All of the bins contain 600 frames each.

3.1.2.3 Dataset_v2

For each recording, the recording was divided into 12 equally sized bins, one bin was discarded from the beginning of the recording and at the end to avoid sampling frames that does not contain the subject or frames that contain initial setup calibration. This left us with 10 bins for each recording. We then assigned a camera to each bin, rotating over all available cameras until each bin had one camera assigned. For each bin, we picked the middle frame from the assigned camera. This approach maximizes the likelihood that at least one sampled frame contain the relevant information.

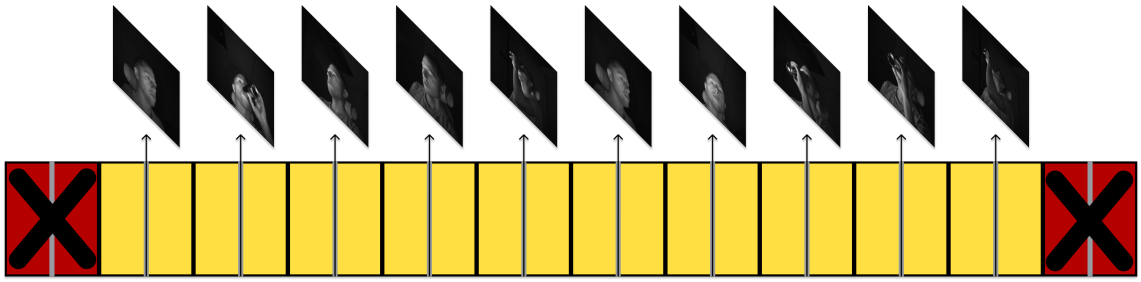


Figure 3.2: Illustration of the frame sampling process for dataset_v2, visualizing the rotation over cameras and discarding the first and last bin.

3.1.3 Data Preprocessing

The dataset used in this study contains images of varying sizes and aspect ratios. To avoid changing the aspect ratio of the images and potentially making it harder for the model to learn, all images were resized and padded to a standardized dimension of 224 x 224 pixels. Additionally this prevents any loss of important information due to cropping or distortion of aspect ratio. Specifically, the images were padded with black pixels to create a square of the desired dimensions.

All images are rescaled to be in the range $[-1, 127.0/128]$, This range instead of $[-1, 1]$ is to accommodate for an possible quantization process of models involving signed integers. The purpose behind this rescaling is to standardize the pixel values and bring them within a specific range that is suitable for subsequent processing. Specifically, the lower bound of -1 is used to center the data around zero, while the upper bound of $\frac{127}{128}$ (which is slightly less than 1) ensures that the pixel values are non-negative and scaled appropriately.

The images were also standardized to have a mean around zero and a standard deviation of 0.5. The standardization process involves subtracting the mean value of the image dataset from each pixel and then dividing by the standard deviation. This operation ensures that the pixel values have a mean of zero and a standard deviation of 0.5. Standardizing the images in this manner helps in normalizing the data and reducing the impact of varying pixel intensity ranges across different images, enabling more effective training of the model [29].

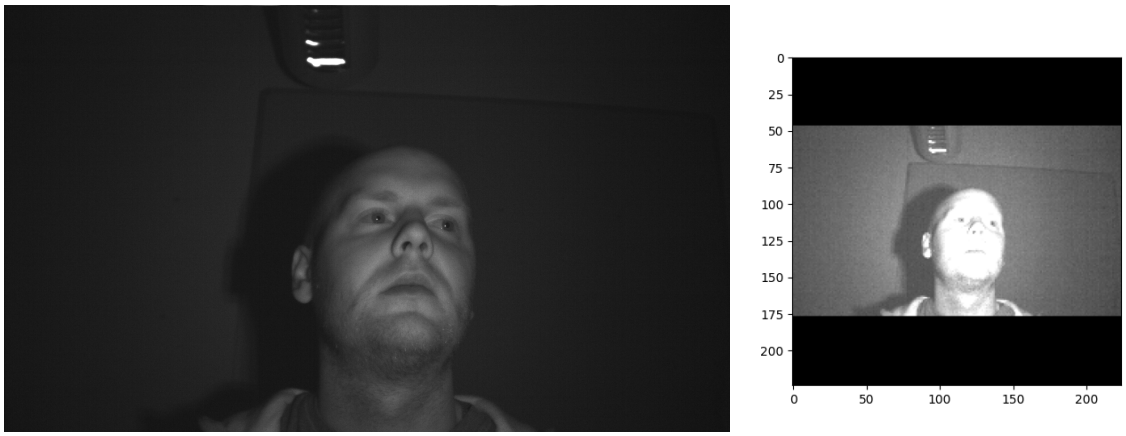


Figure 3.3: The figure is showing the preprocessing step. To the left is the sampled image, and to the right is the preprocessed image.

All images in the dataset were rotated four times, at angles of 0°, 90°, 180°, and 270°. This approach not only increases the amount of training data available, but also allows for the creation of self-supervised labels during training. By rotating the images at these specific angles, it is possible to create new labels for the images, which can then be used to further improve the accuracy of the model. Additionally, this technique has been shown to be effective in reducing overfitting and improving the robustness of deep learning models, particularly in scenarios where the available training data is limited [30].

3.2 Learning Technique

The learning technique utilized in this thesis is based on the approach proposed in [2], but has been modified to better suit the specific needs of our research. The training process consists of three steps, with each step consisting of 10 epochs of training. In contrast to [2], the training process in this thesis utilizes a different number of epochs but also in total the number of training steps per epoch differs since we have different dataset sizes.

A further modification is that each batch contains an unspecified quantity of labeled examples. This is because we are dealing with multiple labels simultaneously, and therefore, we cannot split them into separate data loaders and obtain one batch with unlabeled examples and another with labeled examples during each training step.

During training, we used the ADAM optimizer with specific hyperparameters, including a learning rate of 0.001 and a Cosine Decay Restarts approach. This approach involves setting an initial learning rate and progressively decreasing it over time, based on a cosine function. We used a batch size of 128 for the single-image model (Section 3.3.1) and a batch size of 12 for the multi-image model (Section 3.3.2).

3.2.1 Step 1

In the first step, the model computes the VAT + EntMin loss for each image in every batch and epoch. If an image is labeled for a specific class, a CCE loss is also applied. Additionally, a rotation loss is imposed on all images, requiring the model to predict the rotation of each image through data augmentation. These losses, namely VAT, ENT-min, and rotation loss, are considered self-supervised losses. The overall loss for each label can be expressed as:

$$\text{loss} = 0.3 \cdot L_{\text{sup}} + 0.3 \cdot L_{\text{EntMin}} + 0.3 \cdot L_{\text{VAT}} \tag{3.2}$$

Where L_{sup} is set to 0 if the image does not have a label for that class. Consequently, each image can contribute to a total of # labels * 3 + 1 losses.

3.2.2 Step 2

In the second training step, the process is largely identical to the first step, but with one addition. Before training, the best performing model (i.e., the best epoch), determined by its performance on the validation set, is utilized to generate pseudo labels for all images that have missing labels. This approach allows all images to be treated as labeled data during training, enabling semi-supervised learning. Thus, the second step introduces the concept of semi-supervised learning into the training process. Apart from this modification, the second step follows the same procedures as the first step.

3.2.3 Step 3

In the final step, the model is fine-tuned using only the labeled data without pseudo labels. During this step, only a supervised loss is applied, focusing solely on the labeled samples. The learning rate was decreased to $5 * 10^{-4}$ with 10 folds decay. This fine-tuning process aims to further refine the model's performance using reliable ground truth labels.

3.3 Models

In this thesis, two different architectures were constructed, both based on a scaled-down MobileNetV2 [31]. The primary distinction between the two models lies in their input requirements: the first model accepts only one image as input, while the second model processes 10 images extracted from the same recording. The decision to create the secondary model, which incorporates multiple images, stemmed from the observation that randomly sampled images from a recording might not always capture the subject entirely. Factors such as occlusion by a hand or the steering wheel can obstruct the recording subject. The underlying concept was that by sampling 10 images throughout the recording, as described in Section 3.1.2.3, the model would learn that a single image may lack relevant information.

3.3.1 Single-Image Model

The single-image model, which accepts only one image as input. By utilizing a scaled-down MobileNetV2 architecture, the model aimed to extract meaningful features and make predictions based solely on the provided image. While this approach has its limitations, such as the potential for occlusion or the loss of critical information due to a specific frame selection, it offers simplicity and efficiency in terms of processing time and computational resources. The single image model serves as a baseline for comparison with the multi-image model, allowing us to assess the impact of incorporating multiple frames on the overall performance and accuracy of the system.

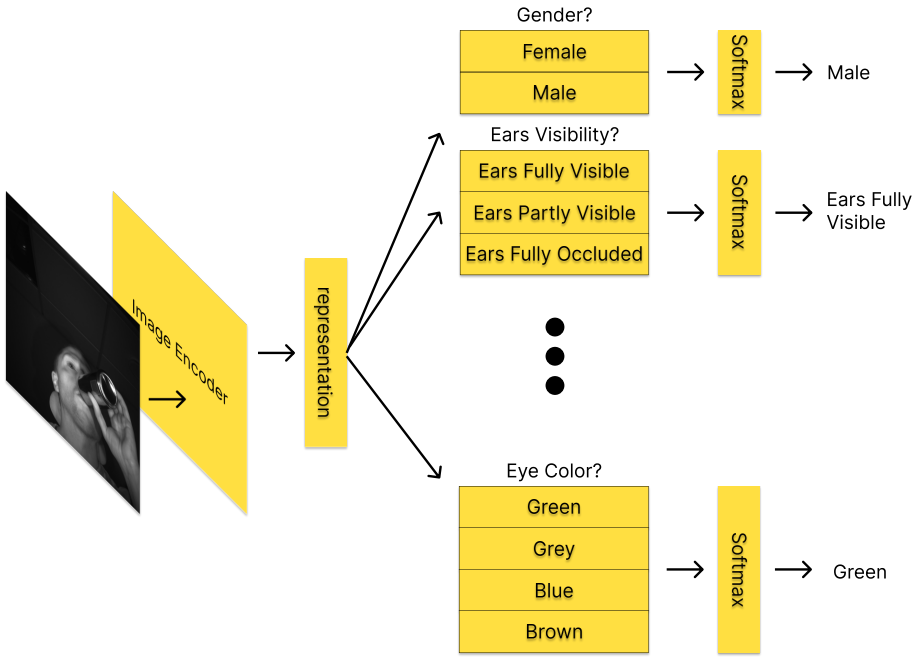


Figure 3.4: An illustration of the architecture for the multi-task single-image model. The different tasks share a common encoding which then branches out to task-specific heads.

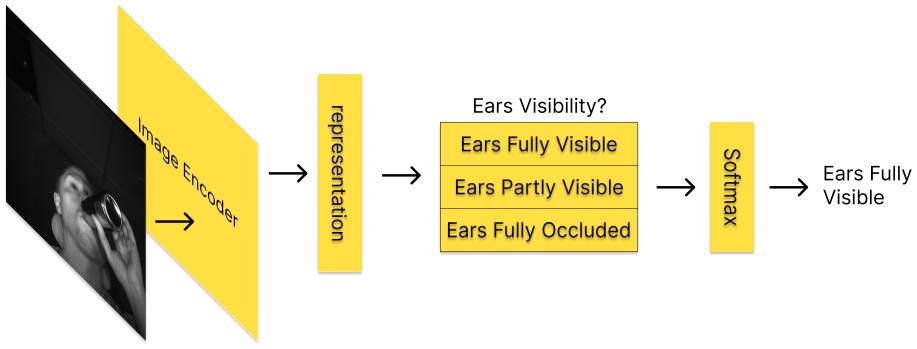


Figure 3.5: An illustration of the architecture for the single-task single-image model. The encoder, outputs a representation for the image which then is used by the classification head.

3.3.2 Multi-Image Model

The multi-image model was developed to overcome the limitations of the single image model by incorporating a sequence of 10 images extracted from the same recording. This decision was driven by the realization that a single image might not always capture the complete context of the subject being recorded. Various factors, such as occlusion caused by objects like a hand or the steering wheel, can obstruct the subject, leading to potential information loss. By sampling 10 images evenly throughout the recording, as outlined in Section 3.1.2.3, the multi-image model aimed to capture a more comprehensive representation of the subject's behavior and environment.

The construction of the multi-image model follows a specific procedure. Each individual image within the sequence is first encoded using the same encoder, which is a scaled-down MobileNetV2. The encoder extracts meaningful features from each image, capturing both low-level and high-level representations. These encoded outputs are then concatenated, combining the information from all 10 images into a single, unified representation. Then additional operations are performed to further process the data.

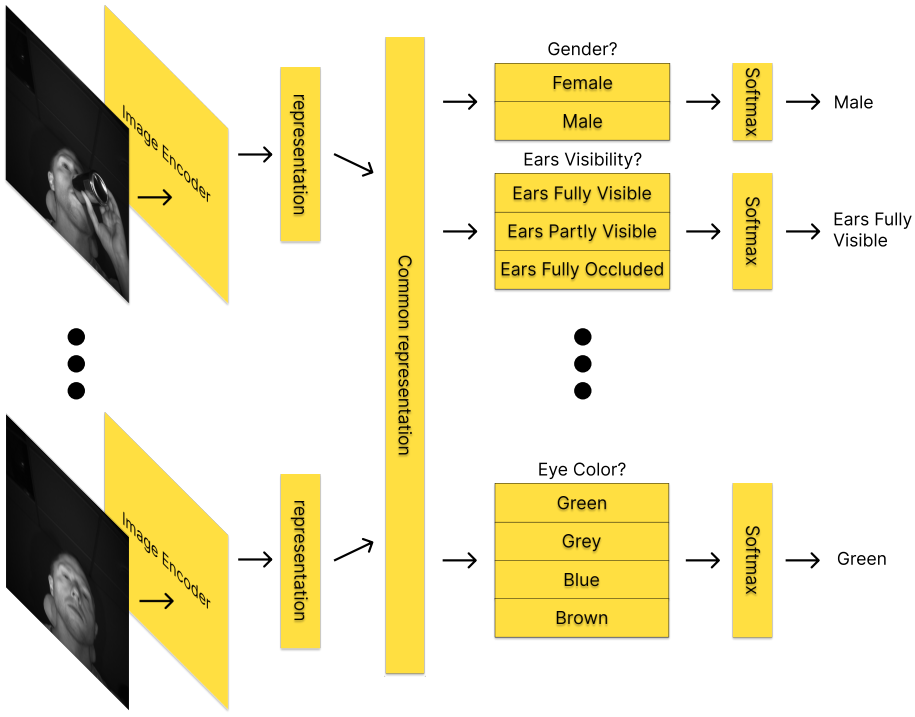


Figure 3.6: An illustration of the architecture for the multi-task multi-image model. Each image is encoded by the same encoder, the encoding is then concatenated into a common representation layer which then branches out to task-specific heads.

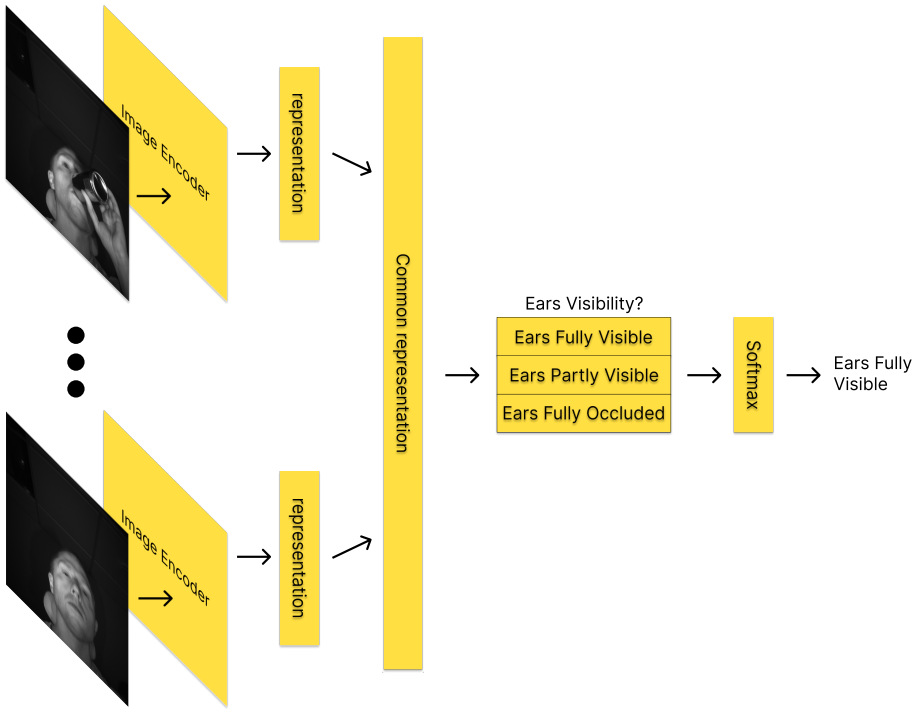


Figure 3.7: An illustration of the architecture for the single-task multi-image model. Each image is encoded by the same encoder, the encoding is then concatenated into a common representation layer which then branches out to the task-specific head.

3.4 Evaluation

To ensure a fair comparison, the finalized model(s) are evaluated on the same test set composed of images from Dataset_v2 (see Section 3.1.2.3). In the case of the single-image model, predictions are made individually for each sampled images from a recording, and the final prediction for the recording is determined by averaging these individual predictions. This approach allows for a direct comparison between the multi-image model and the single-image model, as both the prediction from the single-image model and multi-image model take the same images into consideration.

The evaluation metric used is categorical accuracy, as it primarily focuses on measuring the frequency of correct label predictions. While other metrics can provide additional insights during the model(s) further development, for the purpose of our thesis, we prioritize assessing the overall correctness of the predictions.

3.5 Limitations

The specific setting we are considering is limited to data in the form of near-infrared images of peoples faces. The data is recorded in lab environments or in cars with the purpose of training Smart Eyes driver monitoring system (DMS) which is deployed as a vehicle safety system to assess a drivers alertness and indicate if there seems to be a risk for danger. There are some discrepancies as to where the driver camera is

mounted in different cars and as such there are some discrepancies in the training data, but generally we are considering a chest-up frontal view, or a chest-up side view.

These images are taken from videos where the goal is to label a given video based on a sample of images from that video. The labels that the video receives are metadata labels considering descriptive features of the person driving and the environment that person is in. As such a natural limitation of the model we are going to construct is that it will only be able to predict labels on data that contains all labels that should be predicted and that all labels are able to be identified by a human. For example if we would like to predict the hair-type or the gender of a person it would not make sense to input an image where neither of those features are present.

3.6 Ethical considerations

There are several ethical considerations to keep in mind when working on a data-heavy project such as this one:

1. **Privacy:** It is important to make sure to protect personal and sensitive information. This includes properly securing data storage and making sure to limit access to authorized personnel.
2. **Fairness and non-discrimination:** The data or the models that are developed should not be used to discriminate against or unfairly disadvantage any group of people. It should also not be used to perpetuate or exacerbate existing inequalities.
3. **Transparency:** One should be open and transparent about how the data is collected, used, and shared. This includes disclosing any risks or negative impacts associated with the usage of the data.

Generally it is important to consider the broader context in which our model will be used and the potential consequences of its application.

Our intention is to make advancements in the field of computer vision and potentially free up human resources for Smart Eye. This could allow Smart Eye to more efficiently collect and analyze data, potentially leading to improvements in their machine learning solutions for autonomous vehicles and ADAS. A common worry is that automation leads to unemployment. However according to a report by the UN these claims are often exaggerated [32]. Historically automation rarely substitutes an entire occupation, in most cases it just frees up the people in that occupation to work on other work more suitable for humans (i.e tasks that require more versatility and adaptability). This is in line with goal 8 and goal 9 out of UNs 17 sustainability goals: to promote inclusive and sustainable economic growth, full and productive employment and decent work for all, building resilient infrastructure, promoting inclusive and sustainable industrialization, and foster innovation.

To protect the privacy of individuals depicted in this report, it should be noted that all images have been included with the explicit consent of the persons portrayed.

This means that each individual has provided consent for their images to be published in this report. This measure was taken to ensure that the privacy rights of individuals are respected and that ethical considerations are upheld throughout the research process. Overall, as engineers it is our responsibility to carefully consider the ethical implications of our work and to strive to minimize any negative consequences while maximizing the positive impacts on society.

4

Results

This section goes through the results of the two different model architectures. The models were evaluated on the same test set, this test set was completely unseen data for the models and the people appearing in the test set had not been seen before. There is no publicly available data set that can be used to evaluate and put the results in context. Therefore the test set was put together specifically for this thesis and the reasoning for, and the consequences behind, the construction of this set can be found in Section 3.4.

4.1 Single Image Model

First, we present the results of the Single Image model, as described in Section 3.3.1. Table 4.1 provides the weighted accuracies for both the multi-task learning (MTL) and single-task-learning (STL) variants. The difference between the two is indicated under the Δ column, where a negative value indicates that the STL model outperforms the MTL model. Additionally, the mean difference is denoted as Δ_μ.

Label	**% Labeled**	**MTL Acc.**	**STL Acc.**	Δ
Gender	62.18	76.53	**82.66**	-6.13
Car Recording	8.62	**100.00**	**100.00**	0.00
Ears Visibility	6.34	49.27	**59.42**	-10.15
Chin Visibility	8.42	76.32	**78.01**	-1.69
Glasses Kind	26.10	70.30	**74.17**	-3.87
Mouth Visibility	8.42	92.39	**92.80**	-0.41
Nose Visibility	6.34	66.59	**83.33**	-16.74
Eye Color	38.54	**36.86**	36.54	0.32
Face Mask Kind	29.15	74.89	**78.67**	-3.78
Wearing Glasses	26.10	89.77	**95.60**	-5.83
Wearing Face Mask	29.15	94.62	**95.14**	-0.52
				$\Delta_\mu = -4.44$

Table 4.1: Test accuracies for the different labels. The STL models outperform the MTL model for all labels except Eye Color. The % Labeled column indicates the amount of images that are annotated for the specific label in the training set.

4.2 Multi Image Model

This section presents the results of the Multi Image model, as described in Section 3.3.2. Table 4.2 displays the weighted accuracies for both the multi-task (MTL) and single-task (STL) variants. The difference between the two is indicated using in the Δ column, where a negative value indicates that the STL model outperforms the MTL model. The mean difference is denoted as Δ_μ.

Label	**% Labeled**	**MTL Acc.**	**STL Acc.**	Δ
Gender	60.72	**82.83**	82.03	0.80
Car Recording	10.90	**100.00**	**100.00**	0.00
Ears Visibility	10.72	58.41	**59.35**	-0.94
Chin Visibility	12.84	74.90	**78.91**	-4.01
Glasses Kind	40.43	71.16	**78.01**	-6.85
Mouth Visibility	12.84	93.62	**94.02**	-0.40
Nose Visibility	10.72	82.78	**83.09**	-0.31
Eye Color	42.85	36.01	**38.75**	-2.74
Face Mask Kind	35.52	76.33	**78.26**	-1.93
Wearing Glasses	40.50	92.25	**96.71**	-4.46
Wearing Face Mask	35.52	92.57	**96.66**	-4.09
				$\Delta_\mu = -2.27$

Table 4.2: Test accuracies for different labels. The single-task models outperform the multi-task model for all labels except Gender. % labeled refers to the amount of recordings that are annotated for the specific label in the train set, see Section 3.1.2.3

4.3 Model Comparison

In this section, we present a comparison between the multi-image (M) and single-image (S) models. See Table 4.3

Ears Visibility	M MTL	S MTL	M STL	S STL
Gender	**82.83**	76.53	82.03	82.66
Car Recording	**100.00**	**100.00**	**100.00**	**100.00**
Ears Visibility	58.41	49.27	59.35	**59.42**
Chin Visibility	74.90	76.32	**78.91**	78.01
Glasses Kind	71.16	70.30	**78.01**	74.17
Mouth Visibility	93.62	92.39	**94.02**	92.80
Nose Visibility	82.78	66.59	83.09	**83.33**
Eye Color	36.01	36.86	**38.75**	36.54
Face Mask Kind	76.33	74.89	78.26	**78.67**
Wearing Glasses	92.25	89.77	**96.71**	95.60
Wearing Face Mask	92.57	94.62	**96.66**	95.14

Table 4.3: Test accuracies for both type of architectures. M denotes the Multi Image Model, and S denotes the average predictions for the Single-Image model.

4.4 Individual Labels

In the following sections, the individual class performances are presented for all models.

4.4.1 Gender

Gender	% Labeled	M MTL	S MTL	M STL	S STL
Female	25.85	**77.74**	60.60	72.31	72.31
Male	74.15	87.93	92.46	91.75	**93.02**

Table 4.4: Class Test accuracies for Gender. M denotes the Multi Image Model and S denotes the average predictions for the Single-Image model. % Labeled refers to the relative proportion of the test set.

4.4.2 Car Recording

Car Recording	% Labeled	M MTL	S MTL	M STL	S STL
In Lab	58.41	**100.00**	**100.00**	**100.00**	**100.00**
In Car	41.59	**100.00**	**100.00**	**100.00**	**100.00**

Table 4.5: Class Test accuracies for Car Recording. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of test set.

4.4.3 Ears Visibility

Ears Visibility	% Labeled	M MTL	S MTL	M STL	S STL
Ears Partly Visible	24.57	56.05	54.14	43.95	**68.15**
Ears Fully Visible	63.85	80.00	74.75	**89.50**	65.50
Ears Fully Occluded	11.58	39.19	18.92	**44.59**	**44.59**

Table 4.6: Class Test accuracies for Ears Visibility. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.4 Chin Visibility

Chin Visibility	% Labeled	M MTL	S MTL	M STL	S STL
Chin Fully Occluded	31.94	**80.17**	70.25	78.51	77.69
Chin Fully Visible	68.06	69.63	82.40	**79.30**	78.34

Table 4.7: Class Test accuracies for Chin Visibility. M denotes the Multi Image Model, and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.5 Glasses Kind

Glasses Kind	% Labeled	M MTL	S MTL	M STL	S STL
No Glasses	70.94	87.94	75.06	**93.50**	85.64
Transparent	23.37	59.82	71.96	**72.68**	71.43
Semi-Transparent	3.58	66.28	61.63	**67.44**	65.12
Blocking	2.12	70.59	72.55	**78.43**	74.51

Table 4.8: Class Test accuracies for Glasses Kind. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.6 Mouth Visibility

Mouth Visibility	% Labeled	M MTL	S MTL	M STL	S STL
Mouth Fully Occluded	16.95	**91.54**	89.23	**91.54**	88.46
Mouth Fully Visible	83.05	95.71	95.55	96.50	**97.14**

Table 4.9: Class Test accuracies for Mouth Visibility. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.7 Nose Visibility

Nose Visibility	% Labeled	M MTL	S MTL	M STL	S STL
Nose Fully Visible	99.53	98.89	99.84	99.52	**100.00**
Nose Fully Occluded	0.47	**66.67**	33.33	**66.67**	**66.67**

Table 4.10: Class Test accuracies for Nose Visibility. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.8 Eye Color

Eye Color	% Labeled	M MTL	S MTL	M STL	S STL
Gray	6.33	**9.94**	0.00	3.11	4.35
Green	12.98	**21.65**	2.44	**21.65**	12.80
Blue	38.55	56.25	**71.52**	62.50	63.01
Brown	42.13	56.20	**73.50**	67.76	65.98

Table 4.11: Class Test accuracies for Eye Color. M denotes the Multi Image Model, and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.9 Face Mask Kind

Face Mask Kind	% Labeled	M MTL	S MTL	M STL	S STL
Face Mask Kind None	92.21	95.65	98.14	94.10	**98.45**
Face Mask Kind White	4.33	68.13	**81.32**	79.12	74.73
Face Mask Kind Black	2.80	**91.53**	**91.53**	89.83	**91.53**
Face Mask Kind RGB	0.67	**50.00**	28.57	**50.00**	**50.00**

Table 4.12: Class Test accuracies for Face Mask Kind. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.10 Wearing Glasses

Wearing Glasse	% Labeled	M MTL	S MTL	M STL	S STL
Wears No Glasses	70.94	93.68	**97.34**	96.87	94.50
Wears Glasses	29.06	90.82	82.21	96.56	**96.70**

Table 4.13: Class Test accuracies for Wearing Face Mask. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.4.11 Wearing Face Mask

Wearing Face Mask	% Labeled	M MTL	S MTL	M STL	S STL
Wears No Face Mask	92.21	96.12	97.78	96.38	**99.43**
Wears Face Mask	7.79	89.02	91.46	**96.95**	90.85

Table 4.14: Class Test accuracies for Wearing Face Mask. M denotes the Multi Image Model and S denotes the average predictions for the Single Image model. % Labeled refers to the relative proportion of the test set.

4.5 Contribution of each step

To verify that each step of the method contributes to the overall accuracy of the model, the best performing epoch based on the val set performance of the single-image MTL model was evaluated on the test set after each step.

Model	Step 1	Step 2	Step 3
Gender	72.23	75.94	**76.53**
Car Recording	99.49	96.32	**100.00**
Ears Visibility	**53.58**	42.01	49.27
Chin Visibility	66.70	59.14	**76.32**
Glasses Kind	62.31	61.38	**70.30**
Mouth Visibility	86.65	92.15	**92.39**
Nose Visibility	83.17	**83.25**	66.59
Eye Color	31.87	33.23	**36.86**
Face Mask Kind	63.61	60.49	**74.89**
Wearing Glasses	88.48	78.69	**89.77**
Wearing Face Mask	90.51	93.98	**94.62**

Table 4.15: Weighted test accuracies for the steps of the training process for Single-image MTL model, described in Section 3.2.

5

Discussion

The discussion chapter is divided into multiple sections. First, we have the Data Section, which provides the necessary context for interpreting the results discussed in the Section 5.2. Finally a discussion is had about the comparison between multi-task learning (MTL) and single-task learning (STL).

5.1 Data

There are certain problems that can hinder the learning process and make it more challenging for the model to effectively learn from the data. In this section, we will discuss some of these problematic issues associated with the data that is being dealt with.

5.1.1 Lack of Label Definitions

During the initial phase of the project it was noted that there is no standard being used to set any of the metadata labels. When a new recording is recorded it is up to the people responsible for that particular recording to set the metadata labels they are interested in based on their subjective assessment of what is true and applicable. Although some labels are easy to agree upon, the lack of a standardized definition inevitably leads to ambiguity for some of the labels we are working with. An example is the labels concerned with the visibility of certain facial features in the recording, specifically the chin, the ears, and the nose. Does the chin still count as visible if the subject's head is turned? Would hair covering the ears affect their full visibility? What if only one ear is visible? Should a partially-covered nose due to a face mask still be considered visible?

The issue described above negatively impacts the model's performance by introducing noise into some of the labels. A label is considered noisy when it contains errors, inconsistencies, or ambiguity. When similar cases are labeled differently, the model has difficulty detecting patterns in the data, resulting in inconsistent and unreliable predictions. Additionally this might affect the evaluation of the models since there might be noisy labels present in the validation set and the test set.

5.1.2 Frame Sampling

The use of video frames as training data for the models further exacerbates the problem of noisy labels. Note that the metadata labels are set for complete recordings, not for specific frames. So for instance, if a person in a recording is turning the steering wheel in a way that obstructs their face, then if we sample that frame we would end up with a faulty label. Another example would be that a person is turning their head in a way which occludes a feature that has been labeled as visible. These kinds of examples are hard to work around since they are unpredictable in the sense that they could occur anywhere in the video and no preprocessing can be done to overcome them. This obviously also contributes to the level of noise present in the dataset.

(a) Looking away. (b) Steering wheel occlusion.

Figure 5.1: Two sampled frames illustrating the problem of sampling a frame from a recording. The chin is annotated as Fully Visible in both these examples.

5.2 Results

The discussion of the result is structured into multiple sections. The first three sections will provide a general discussion about the single-image model, the multi-image model, and then a combined discussion of both model types. Subsequently, a more in-depth discussion will be conducted for each label.

5.2.1 Single Image model

The single-image model, shows promising results for both MTL and STL. Car Recording, Wearing Face Mask, Wearing Glasses and Mouth Visibility achieve accuracies almost over 90% when looking at the weighted accuracies for the labels in Table 4.1.

Worth noting is that the biggest delta difference between MTL and STL comes from Ears Visibility and Nose Visibility, and without these the mean difference would be $\delta_\mu = -2.43$. Both these labels are bad performers when looking at the weighted accuracies and the reasons for this will be discussed in their respective section, see

Section 5.2.7 and 5.2.11. Overall the MTL model seems to be almost on par for most of the best performing labels.

5.2.2 Multi Image model

Overall, the multi-image model shows promising results across various labels for both the single-task and multi-task versions. Notably, Car Recording, Mouth Visibility, Wearing Glasses, and Wearing Face Mask achieve accuracies of over 90% for both MTL and STL. The only label which the multi-task model outperforms the single-task model is the Gender label, where it achieves an accuracy that is 0.80 percentage points higher.

Comparing the performance of the single-task and multi-task versions, both models achieve similar overall accuracy. However, it is important to note that the multi-task model's average accuracy across all labels is slightly lower compared to the single-task models, as indicated by the Δ_μ value of -2.27. This suggests that further analysis and optimization may be required to enhance the multi-task model's performance.

5.2.3 Combined discussion

When examining Table 4.3 M STL looks like the clear winner, since it is the model which has the best weighted accuracy for most labels. However, the other models are not that far behind performance wise.

We can also see that the label wise performance difference is smaller for the multi-image model type than for the single-image model type when looking at their corresponding Tables 4.1 and 4.2. This is probably a sign that the multi-image model suffer less from the frame sampling problem which was discussed earlier.

5.2.4 Individual Labels

In this section the results of the individual labels is discussed.

5.2.5 Gender

The best performing model on the Gender label is the multi-image MTL model which has a weighted accuracy of 82.83% as can bee seen in Table 4.3. By examining Table 4.4, it becomes apparent that all models perform better on predicting males, with the multi-image MTL model displaying the least discriminatory behavior. Several factors may contribute to this, and one hypothesis is that the multi-image MTL model's capacity to learn from multiple images and tasks simultaneously allows for a more comprehensive understanding of gender characteristics.

Another possible explanation could be the presence of a higher number of labeled examples for males in our data subset. This highlights the importance of training models on a diverse range of images. Although we address imbalances through

weighting techniques, a larger representation of male images may still contribute to their higher accuracy scores.

5.2.6 Car recording

All models excel at predicting both if the recording is recorded in a lab environment or in a car as can be seen in Table 4.5. Interestingly enough Car Recording is one of the labels that have the fewest amount of labels with only approximately 10% of the recordings.

A possible explanation for these results is that for this label there should be almost no occlusion that makes it impossible to extract the necessary information of whether or not the recording took place in a car.

The results of Car Recording shows the potential for the learning framework assuming the labeling is done in a consistent manner where there is little to no noise and no ambiguity.

5.2.7 Ears Visibility

Ears Visibility stands out as one of the poorest performing labels, alongside Eye Color. We specifically explored the adoption of the multi-image model because of the poor initial results of Ears Visibility, due to the inherent ambiguity and lack of precise definition in the label categories. For instance, consider Figure 5.2, should the Ears Visibility label associated with the recording from which the images are sampled be annotated as Partly Visible or Fully Occluded?

(a) Partly visible ear. (b) Fully occluded ear.

Figure 5.2: Two images illustrating the difficulties the model faces due to the ambiguity of Ears Visibility, should the recording be labeled as partly visible or fully occluded?

It is worth noting that Ears Visibility has a significantly lower number of annotated recordings compared to gender which has the most labeled recordings. With only 10.72% of recordings in the test set that are annotated for Ears Visibility, it poses a

challenge for the model's learning and generalization ability compared to the more extensively annotated labels. This in addition to the ambiguity and noise in annotation contributes to the complexity and potential limitations of accurately predicting Ears Visibility.

Upon examining the results presented in Table 4.6, it becomes evident that the model encounters difficulties in learning and accurately predicting both the Partly Visible and Fully Occluded classes. The accuracies for these categories, particularly Partly Visible, are notably lower compared to the other label categories. This highlights the difficulty in distinguishing and categorizing different levels of ear visibility within our dataset.

One interesting finding is that the multi-image MTL models appear to benefit significantly from considering a batch of 10 images from the same recording simultaneously, as opposed to the single-image MTL model's approach of averaging predictions. This observation demonstrates a substantial improvement in performance, which supports our hypothesis that considering more images lead to better performance.

5.2.8 Chin Visibility

Chin Visibility achieves a score of above 80% for the best performing model, which is the single-image STL model with an accuracy of 82.66%, as shown in Table 4.7. It is worth noting that, similar to Ears Visibility, Chin Visibility has a relatively limited number of annotated recordings in the training set. Only 12.84% of the recordings have this label set, indicating a scarcity of data for this specific attribute. This scarcity of annotated samples poses a challenge for the model's training and may impact its ability to accurately predict Chin Visibility.

Upon analyzing Table 4.7, an interesting observation can be made regarding the performance of the single-task models. Both the fully occluded and fully visible classes appear to present similar difficulties for these models in terms of accurate prediction. However, a noteworthy pattern emerges when examining the multi-task learning versions. The multi-image model demonstrates superior performance in predicting the fully occluded category, while the single-image model excels in predicting the fully visible category. This contrast highlights the potential benefits of utilizing the multi-image model for capturing subtle visual cues and contextual information relevant to fully occluded chins, while the single-image model may excel at discerning features indicative of fully visible chins.

5.2.9 Glasses Kind

Glasses Kind was one of the noisier labels, perhaps due to the fact that some classes are rather ambiguous. In the case of Glasses Kind case there are no problems with occlusions due to frame sampling, however there are a lot of cases where the type of glasses seem to be similar but are labeled differently. There are even some cases where the exact same type of glasses are labeled differently in different recordings. The problem seems to be that there are no clear definition stating what makes a pair of glasses transparent, rather than semi-transparent. This is apparent in Table

4.8 where Transparent and Semi-Transparent are the worst performing classes for all models.

To combat this noise loss functions were used that punish low entropy output distributions, this prevents overfitting on noisy label by smoothing the labels. This is the opposite of what EntMin tries to accomplish since EntMin punishes high entropy output distributions, but it has been shown that penalizing low entropy distributions acts as a strong regularizer [33]. This is not something that has been explored during this thesis, but the experimentation of additional loss functions could be an interesting path to pursue to combat the noise present in some labels.

5.2.10 Mouth Visibility

All models perform relatively well on the Mouth Visibility label with accuracies ranging from approximately 90% up to 97%. This label should be relatively unambiguous to annotate since there are only two classes: Fully Occluded and Fully Visible. However this label is affected by the frame sampling problem illustrated in 5.1, as in both images the class is Mouth Fully Visible, but the mouth is not visible in any of the images. This is a clear example where considering more images should be beneficial since this reduces the impact of eventual occlusions, and we do in fact see that the multi-image models perform marginally better (roughly 1-2%).

5.2.11 Nose Visibility

Similar to the Mouth Visibility label Nose Visibility is a label that all models perform relatively well on, however there is a clear discrepancy between the MTL model and the STL model in the single-image model case, as can be seen in Table 4.1. The STL model outperforms the MTL model with > 15 percentage points. However looking at Table 4.10 we see that the big difference manifests is in how accurately the models predict the Nose Fully Occluded class; in the Nose Fully Visible class the models have very similar performance. This is an issue that stems from the fact that there is a very large class imbalance present for this label which means that just getting a few examples wrong during the evaluation will result in much worse performance due to weighting.

5.2.12 Eye Color

Eye Color was a label that was chosen to be included in the model due to it being an interesting case study if the MTL aspect would somehow increase performance for predicting the eye color of the person. Since the data format is NIR there is also no presence of any color in any of the images, so if the performance is better than guessing that is also interesting. There are however differences that can be observed even in the absence of color, such as the brown eyes being noticeably darker than blue/gray eyes.

It can be seen in Table 4.3 that the MTL did in fact not outperform the STL model in neither the multi-image or single-image case. However the performance is very

similar in the mid 30% range. This is better than randomly guessing since there are 4 classes present so randomly guessing would result in an accuracy of 25% if the classes were balanced. However the classes are not balanced so there is probably some majority class bias going on towards the brown and blue eyes which both have a lot higher accuracies than gray and green eyes.

5.2.13 Face Mask Kind

Face Mask Kind and Glasses Kind are similar in the sense that they are trying to predict what type of facial accessory a person is wearing. They also exhibit similar accuracies ranging from the low to high 70s. However looking at Table 4.12 we see that predicting whether a person is wearing no face mask, or if they are wearing a black face mask, is very easy for all the models, ranging to the high 90s in accuracy for the best performing single-image STL. Similar to the eye-color case we have a class imbalance caused by a relatively low presence of Face Maks Kind RGB which influences the weighted accuracies of Face Mask Kind for the models. This label also seems to have some ambiguity problems since it has been seen that some of the face masks labeled as RGB are either black or white, which leads to confusion for the models.

5.2.14 Wearing Glasses

The Wearing Glasses label seems to be an easy label for all the models with accuracies ranging from low to high 90s as can be seen in Table 4.13. The two different classes also exhibit similar performance which is a good sign that the weighting works or that the underlying class distribution is similar. Interestingly the No Glasses class from Glasses Kind has a slightly worse performance than Wearing No Glasses, confirming the claim that it is indeed easier to just predict whether a person is wearing glasses, and not the type of glasses that they are wearing.

5.2.15 Wearing Face Mask

Similar to Wearing Glasses, Wearing Face Mask has high performance for all models, with the best model (single-image STL) almost achieving an accuracy of 100% for the class Wearing No Face Mask, which is similiar to the top performer for the class Face Mask Kind None, for the Face Mask Kind label. However this label is not weighted down by any class imbalances which explains the higher weighted accuracy. This is another example of a label being reduced to a simpler label containing less information, but being easier for the models to predict, similar to Wearing Glasses.

5.3 Contribution of Each Step

The test performance of the single-image MTL model is demonstrated in Table 4.15, showcasing improvements for most labels after each step. Notably, Ears Visibility and Nose Visibility emerge as two classes that stand out. This observation can be attributed to the factors discussed earlier in their respective sections. For Nose

Visibility, the presence of a small portion of Nose Fully Occluded results in minor changes in correct predictions, significantly affecting the weighted accuracy of the label. On the other hand, Ears Visibility highlights the label's instability, likely stemming from data noise and inconsistencies during the labeling process. Although the extent of improvement may vary across labels, it is worth mentioning that almost all labels demonstrate their highest performance on the test set after Step 3.

5.4 Multi-Task vs Single-Task

Multi-task learning provides several advantages over single-task learning, irrespective of the accuracy metric. One key benefit is the simplified maintenance of a single model. Instead of dealing with separate models for each task, multi-task learning enables training a single model to handle multiple tasks simultaneously.

Another advantage of multi-task learning is the utilization of shared weights among tasks. By sharing specific layers or parameters across tasks, the model can exploit commonalities and benefit from the knowledge acquired during the training process. This weight sharing leads to reduced computational requirements during inference, which becomes particularly advantageous when working with limited hardware resources.

Additionally, multi-task learning results in significantly reduced training times due to the decreased number of parameters to be trained. Unlike single-task learning, where each task requires a separate model with its own set of parameters, multi-task learning shares parameters among tasks. As a result, the overall number of trainable parameters decreases, leading to faster training times and improved training efficiency.

Our experiments show that for most labels multi-task is not lacking that much behind even though some labels do not have any clear relationships to the other labels, for example Mouth Visibility and Car Recording. This gap can most likely be reduced or even completely removed by trying more model architectures.

To summarize, multi-task learning offers benefits beyond accuracy considerations, including simplified model maintenance, reduced computational requirements during inference, and decreased training times due to the smaller parameter count. These advantages make multi-task learning advantageous, particularly when dealing with limited hardware resources or resource constraints.

6

Conclusion

In conclusion, this thesis has addressed the emerging research area of self-supervision and semi-supervision, specifically focusing on the challenges posed by multi-class scenarios. While previous studies primarily focused on single-task models trained on datasets with single-class labels, this research has addressed the challenges of multi-class scenarios, which can involve partially labeled data where not all datapoints have all labels defined.

We propose a modified S4L framework [2] that can be applied to both multi-task learning and single-task learning scenarios, enabling the effective utilization of partially labeled data.

The effectiveness of the modified S4L framework has been demonstrated through training models in both multi-task learning and single-task learning settings. Our research has focused on classifying visual attributes of human subjects in near-infrared (NIR) images. Our experiments have shown the capability of the modified S4L framework to accurately classify visual attributes under these conditions.

Despite the observed lower performance of multi-task learning compared to single-task learning in this specific setting, the potential benefits of multi-task learning may still outweigh the performance loss. However, further research is necessary to conclusively determine whether multi-task learning or single-task learning is the better choice in this setting.

6.1 Proposed Solution

The proposed solution, the multi-image model described in Section 3.3.2, exhibits only a marginal decrease in accuracy compared to the other models. However, we believe that the advantages of reduced training times, computational resources, and maintenance, among others, outweigh this slight performance loss. We also believe that our model has room for improvement, therefore further research would allow for better performance.

Bibliography

[1] T. Fredriksson, D. I. Mattos, J. Bosch, and H. H. Olsson, "Data labeling: An empirical investigation into industrial challenges and mitigation strategies," in *Product-Focused Software Process Improvement*, M. Morisio, M. Torchiano, and A. Jedlitschka, Eds., Cham: Springer International Publishing, 2020, pp. 202–216, ISBN: 978-3-030-64148-1.

[2] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning series). MIT Press, 2016, ISBN: 9780262035613. [Online]. Available: https://books.google.se/books?id=Np9SDQAAQBAJ.

[4] O. Chapelle, B. Schölkopf, and A. Zien, in *Semi-Supervised Learning*. MIT-Press, pp. 1–12.

[5] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, ser. CIKM '05, Bremen, Germany: Association for Computing Machinery, 2005, pp. 195–200, ISBN: 1595931406. DOI: 10.1145/1099554.1099591. [Online]. Available: https://doi.org/10.1145/1099554.1099591.

[6] C. M. Bishop, "Neural networks and their applications," *Review of Scientific Instruments*, vol. 65, no. 6, pp. 1803–1832, Jun. 1994. DOI: 10.1063/1.1144830. [Online]. Available: https://doi.org/10.1063/1.1144830.

[7] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. arXiv: 1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458.

[8] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[9] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1979–1993, 2019. DOI: 10.1109/TPAMI.2018.2858821.

[10] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 regularization for learning kernels," *arXiv preprint arXiv:1205.2653*, 2012.

[11] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf.

[12] [Online]. Available: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[13] A. Shafahi, M. Najibi, M. A. Ghiasi, *et al.*, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[14] J. M. Joyce, "Kullback-leibler divergence," in *International encyclopedia of statistical science*, Springer, 2011, pp. 720–722.

[15] T. Nguyen, B. Lyu, P. Ishwar, M. Scheutz, and S. Aeron, "Conditional entropy minimization principle for learning domain invariant representation features," *CoRR*, vol. abs/2201.10460, 2022. arXiv: 2201.10460. [Online]. Available: https://arxiv.org/abs/2201.10460.

[16] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–2. DOI: 10.1109/IWQoS.2018.8624183.

[17] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, Sep. 2017, ISSN: 2095-5138. DOI: 10.1093/nsr/nwx105. eprint: https://academic.oup.com/nsr/article-pdf/5/1/30/31567358/nwx105.pdf. [Online]. Available: https://doi.org/10.1093/nsr/nwx105.

[18] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," *CoRR*, vol. abs/1901.09005, 2019. arXiv: 1901.09005. [Online]. Available: http://arxiv.org/abs/1901.09005.

[19] A. Zbiciak and T. Markiewicz, "A new extraordinary means of appeal in the polish criminal procedure: The basic principles of a fair trial and a complaint against a cassatory judgment," en, *Access to Justice in Eastern Europe*, vol. 6, no. 2, pp. 1–18, Mar. 2023.

[20] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[21] A. Kurakin, C. Raffel, D. Berthelot, *et al.*, "Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring," in *ICLR*, 2020. [Online]. Available: https://openreview.net/pdf?id=HklkeR4KPB.

[22] X. Wang, D. Kihara, J. Luo, and G.-J. Qi, "Enaet: A self-trained framework for semi-supervised and supervised learning with ensemble transformations," *IEEE Transactions on Image Processing*, vol. 30, pp. 1639–1647, 2021. DOI: 10.1109/TIP.2020.3044220.

[23] T. Han, J. Gao, Y. Yuan, and Q. Wang, "Unsupervised semantic aggregation and deformable template matching for semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9972–9982, 2020.

[24] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[25] E. Wallin, L. Svensson, F. Kahl, and L. Hammarstrand, "Doublematch: Improving semi-supervised learning with self-supervision," *arXiv preprint arXiv:2205.05575*, 2022.

[26] J. Kleinberg and E. Tardos, *Algorithm Design*. Upper Saddle River, NJ: Pearson, Jun. 2005.

[27] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.
[28] R. E. Gomory, "Solving linear programming problems in integers," *Combinatorial Analysis*, vol. 10, pp. 211–215, 1960.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[30] C. Khosla and B. S. Saini, "Enhancing performance of deep learning models with different data augmentation techniques: A survey," in *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, IEEE, 2020, pp. 79–85.
[31] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. arXiv: 1801.04381. [Online]. Available: http://arxiv.org/abs/1801.04381.
[32] *Frontier issues: The impact of the technological revolution on labour markets and income distribution | department of economic and social affairs*. [Online]. Available: https://www.un.org/development/desa/dpad/publication/frontier-issues-artificial-intelligence-and-other-technologies-will-define-the-future-of-jobs-and-incomes/.
[33] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. E. Hinton, "Regularizing neural networks by penalizing confident output distributions," *CoRR*, vol. abs/1701.06548, 2017. arXiv: 1701.06548. [Online]. Available: http://arxiv.org/abs/1701.06548.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY