A Hybrid Approach to Melanoma Classification

Improving medical diagnosis by combining machine learning with human expertise

Master's Thesis in Data Science - Spring 2022

Student:
Frederik Rasmus Andersen frean11

Counsellors:
Pantelis Pipergias Analytis Primary
Richard Röttger Co-counsellor

University of Southern Denmark
Department of Mathematics and Computer Science

Code published on Github
github.com/frean11/Masters_Thesis

Dansk resumé

Baggrund: Der kan reddes liv ved at øge præcisionen når medicinske professionelle evaluerer risikoen for kræft i modersmærker. Nylig udvikling i CNN netværk har gjort dem mindst lige så gode som mennesker til at diagnosticere hudkræft og man har set lovende resultater fra hybrid tilgange som kombinerer mennesker og computere. Jeg præsenterer i dette speciale to hybrid algorithmer som forsøger at kombinere evnerne hos mennesker med et CNN på en måde som er relativt billig i forhold til menneskelige ressourcer.

Metoder: Jeg træner et CNN, som bruger EfficientNetB3 modellen som rygrad, på data fra 2019 og 2020 ISIC konkurrencerne. Denne model lader jeg forudsige 150 billeder hvor jeg allerede har forudsigelser fra 69 medicinske professionelle. Svarene fra mennesker og CNN bruges til at teste de to hybrid algoritmer. Alle resultater bliver evalueret med ROC/AUC analyse og de menneskilige ressource omkostninger associeret med hver tilgang bliver beregnet.

Resultater: Min reference CNN opnår en AUC score på 0.822. Til sammenligning med den menneskelige referenceværdi, ved samme FPR har menneskene TPR = 0.765 imens CNN har TPR = 0.694 og ved samme TPR har menneskene FPR = 0.196 imens CNN har FPR = 0.320. Dette indikerer at CNN er dårligere end den menneskelige reference.
Sammenligning mellem min augmented hybrid algoritme og den menneskelige referenceværdi, ved samme FPR har algorithmen TPR = 0.782 og ved samme TPR har algoritmen FPR = 0.182. Dette indikerer algoritmen er en anelse bedre end den menneskelige reference og er opnået med kun et menneske pr billede. Min hybrid majority vote algoritme viser sig at være bedre end den menneskelige referenceværdi og sammenlignes med et ensemble bestående af tre mennesker. Ved lignende FPR værdier har det menneskelige ensemble TPR = 0.787 og algoritmen har TPR = 0.765. Ved lignende TPR værdi har det menneskelige ensemble FPR = 0.150 og algoritmen har FPR = 0.169. Så hybrid majority vote algoritmen er en anelse dårligere end et tre menneskes ensemble, men til gengæld kræver den færre mennesker pr billede (1.26-1.53) i forhold til et ensemble med tre mennesker (2.24).

Begrænsninger: Dette studie er begrænset af en dårlig performance på mit reference CNN. Ifølge litteraturen burde CNN være bedre end mennesker til denne type opgave, men det er ikke tilfældet for dette studie. Sammenligning af reference netværket med stillingen på den Kaggle konkurrence som trænings datasættet kommer fra viser at modellen ligger på den nederst 37. procentdel. Jeg tror resultatet vil generalisere med bedre CNN modeller, men dette bør verificeres gennem testning med andre datasæt og bedre modeller.

Konklusion: Begge algorither præsenteret i dette speciale er billige måder at forbedre præcisionen i diagnoser hos medicinske professionelle, hvor det stadig er mennesker, som har den endelige beslutning. Der skal dog forskes mere i algoritmerne med bedre modeller for at sikre at de generaliserer.

English abstract

Background: Increasing the diagnosis performance on malignant skin lesions in medical offices can save lives. Recent development in CNN performance has made them at least at good as human in these diagnosis tasks and various hybrid approaches which combine humans and computers have shown promising results. I present in this thesis two hybrid algorithms which seek to combine the strengths of humans and a CNN while being cheap in terms of the human resources necessary.

Method: I train a CNN using an EfficientNetB3 backbone on a data set from the 2019 and 2020 ISIC Challenges. This model predicts on a 150 image data set to which I know the predictions from 69 trained medical professionals. The human and CNN predictions are used to test two hybrid intelligence algorithms. All results are evaluated using ROC/AUC analysis and the human resource cost associated with each approach is calculated.

Findings: The baseline CNN performance achieves an AUC score of 0.822. Comparing to human baseline, at similar FPR rates the average human has TPR = 0.765 while the CNN has TPR = 0.694. At similar TPR rates, the average human has FPR = 0.196 while the CNN has FPR = 0.320. This means the CNN is inferior to the human baseline.
Comparing the augmented hybrid approach to human baseline, at similar FPR rates the algorithm achieves TPR = 0.782. At similar TPR rates, the algorithm achieves FPR = 0.182. This is slightly better than human average and is achieved with minimal human resource cost of 1.
The hybrid majority vote approach is superior to human baseline and because it is an ensemble, a more fair comparison is against a three-human ensemble. At similar FPR rates, the human ensemble achieves TPR = 0.787 and the algorithm achieves TPR = 0.765. At similar TPR rates, the human ensemble achieves FPR = 0.150 and the algorithm achieves FPR = 0.169. The hybrid majority vote performs slightly worse than a three-human ensemble, but it comes with a cheaper average cost (range 1.26-1.53) compared to the three-human ensemble (2.24).

Limitations: This study is limited by a weak performance of the baseline CNN. Contemporary literature shows CNNs should be better than human baseline performances, which is not the case for this thesis. Compared to the Kaggle leaderboard related to the training data set, the baseline CNN performs in the bottom 37^{th} percentile. I theorize the results will generalize with stronger CNN models, but this should to be verified through additional testing with other data sets and SOTA networks.

Conclusion: The algorithms presented are both cheap ways of improving medical diagnosis performance while keeping humans in the loop; however, more research should be done using the algorithms with SOTA CNN models to see if the results generalize.

Contents

1 Introduction

Skin cancer is one of the most common and most aggresive forms of cancer, affecting more than five million people annually in the United States alone (Rogers, Weinstock, Feldman, & Coldiron, 2015). Early and correct diagnosis is key to treatment and many resources are being spend on optimizing diagnotic tools. When reviewing the research, I have found three categories of research. The first category is research into techniques for improving human decision making and thus increase accuracy. This can be either by aggregating opinions (R. Kurvers et al., 2019; R. Kurvers, Herzog, Hertwig, Krause, & Wolf, 2021; R. Kurvers, Krause, Argenziano, Zalaudek, & Wolf, 2015; R. H. J. M. Kurvers et al., 2016) or by developing diagnostic techniques which yield higher accuracies (Zalaudek et al., 2006). Another category of research is Artificial Intelligence (AI), specifically the development of Convolutional Neural Networks (CNN) which classify malignancy from pictures (T. Brinker et al., 2019; T. J. Brinker et al., 2019; Esteva et al., 2017; Ha, Liu, & Liu, 2020; S. Han et al., 2018; Hekler, Utikal, Enk, Berking, et al., 2019; W. Li, Zhuang, Wang, Zhang, & Zheng, 2020; Marchetti et al., 2020; Tschandl et al., 2019). The third category of research tries to combine the first two categories into hybrid models where human and computer computer predictions are aggregated or ensembled to yield even better performance (S. S. Han et al., 2020; Hekler, Utikal, Enk, Hauschild, et al., 2019; Tschandl et al., 2020).

The idea of aggregating opinions has been around for centuries. In 1785 Nicolas Concordet concluded that aggregating the opinions of many humans is expected to give the right answer if each human has probability $p > \frac{1}{2}$ of making the right decision (Boland, 1989). Francois Galton would later claim that an aggregated group opinion would cancel out the individual noise and yield the correct answer (Galton, 1907). Today this principle is known as the "Wisdom of the Crowds phenomenon". In the literature of skin lesion classification, R. Kurvers et al. (2019) has shown the principle to hold under certain circumstances. He proved that aggregating doctors opinions can increase prediction accuracy if the doctors in the ensemble perform similarly. In a three-doctor ensemble where the doctors have similar individual performance, aggregating their opinions in a majority vote will create a higher accuracy than either of the doctors score individually. However, if one doctor is individually far superior to the other doctors, this ensemble will perform worse than the best individual doctor. This observation has profound consequences on the opportunity to create ensemble models between humans and machines.

The merit of Convolutional Neural Networks was segmented in 2012 when a CNN named AlexNet won the ImageNet Classification Challenge (S. Han et al., 2018). A few years later, computer models had surpassed human performance on this visual classification challenge and the algorithms behind these computer vision networks were applied to other fields of visual classification. In two studies, H. Haenssle et al. compared the performance of CNNs on image classification of skin lesions to that of human experts (H. Haenssle et al., 2018; H. A. Haenssle et al., 2020). Both studies revealed computers to be generally superior to humans. This claim is supported by multiple other studies (T. Brinker et al., 2019; T. J. Brinker et al., 2019; Esteva et al., 2017; Hekler, Utikal, Enk, Berking, et al., 2019; W. Li et al., 2020; Marchetti et al., 2020) and a metareview by Haggenmüller et al. (2021, p. 213-214) concluded that: "All 19 included studies demonstrated superior or at least equivalent performance of CNN-based classifiers compared with clinicians". However, they also highlight that most of these comparative studies are conducted under artificial conditions. As mentioned by H. A. Haenssle et al. (2020), the comparative studies often put the clinicians on the computers home court by asking the clinicians to make their predictions from pictures. In their normal practice, dermatologists have the opportunity to look at the skin lesions directly and use available metadata to make their prediction, so having only a picture of the lesion is thought to be a disadvantage for the dermatologist. The authors

of the paper introduce two levels of information - Level I and Level II. They refer to Level I information as the readily available pictures and Level II information as the metadata pertaining to each image. The findings from the study suggests that when clinicians have only Level I information available, the computer algorithms have superior performance. When Level II information is available, the performance is rather similar.

With the strong performance of computers on skin lesion classification tasks, newer research has started looking into the opportunities for combining man and machine into hybrid approaches. The idea being that if ensembles can improve accuracy, then a human-computer combination may be able to outperform both the human and the computer individually. A massive research project by S. S. Han et al. (2020) has shown that not only does AI outperform humans in these classification task, but that using the computer algorithm to aid in human decisions can prove beneficial. They let human subject experts investigate and classify images of skin lesions in the same way as many other studies have, and found that adding the prediction value from a trained CNN as information could positively influence the decision of the human. They termed this procedure "augmented intelligence". The findings are similar to those by Tschandl et al. (2020) who also found that the least experienced clinicians gain the most from the computer prediction. They do however point out that faulty AI can mislead the entire spectrum of clinicians which could pose a potential threat to the useability of the augmented intelligence hybrid approaches. If humans start trusting the AI decisions blindly, the resulting predictions will by those of the AI and not yield the superior hybrid performance which is found in the described studies.
A study by Marchetti et al. (2020) took a slightly more systematic approach and asked their human participants to rate their confidence in each picture. They found that by imputing the estimates on pictures with low confidence with a prediction from a computer algorithm, they were able to improve on performance measures. This effect was most profound in medical residents and less prevalent in professional dermatologist, although still present. This method is still vulnerable to faulty AI, but it does not run the risk of humans relying too heavily on the CNN predictions as they are not available to the human subjects.
Hekler, Utikal, Enk, Hauschild, et al. (2019) studied yet another hybrid approach. They had 112 dermatologists and one trained computer algorithm classify 300 pictures of skin lesions. They then aggregated the prediction using a boosting algorithm and found that the combination approach was best for both multiclass and for binary classification of the lesions. This suggests that an ensemble of human and computer decisions is superior. The approach does not allow humans to rely on the results of a computer and seems to me like a more viable approach for a real-life implementation. However, the presented approaches have not been compared to the best of my knowledge.

This section has presented three main ideas. The first idea is that aggregating opinions is good if the elements of the ensemble have similar performance. This strategy could be used to improve clinical performance, but it comes with a high cost as a majority vote requires asking multiple clinicians for each case. The second idea is that CNNs are at least as good as humans in malignancy classification of skin lesions. However, more research is still needed on this subject and the algorithms have been deemed not yet good enough to overtake the role of classification from humans (Haggenmüller et al., 2021; S. S. Han et al., 2020; Navarrete-Dechent, Liopyris, & Marchetti, 2021). Replacing humans completely with CNNs would require that the algorithmic performance improves further along with much work on the ethical aspects. In the mean time, research into the third idea of hybrid approaches seems like a logical next step in the research. I have presented multiple different hybrid approaches which have been shown to yield better performance than humans or computers can achieve individually. I believe in the potential of hybrid intelligence and think it could save lives. However, I think the literature lacks a discussion of the costs associated with the hybrid approaches, because

a high cost can make even a high performing method implausible to implement. Take for instance the study by Hekler, Utikal, Enk, Hauschild, et al. (2019) which used 112 dermatologists. Having 112 professionals look at the same picture does not seem like an applicable solution in a real world setting. So for this Thesis, I look into creating methods which improve performance while being relatively inexpensive to implement. I create two hybrid algorithms which expands on the current research, are low-cost implementations, and could save lives. The following two sections motivate these algorithms which I refer to as "the augmented hybrid approach" and "the majority vote hybrid approach".

The augmented hybrid approach builds on the principle behind the S. S. Han et al. (2020) which improved prediction by providing a CNN prediction score as metadata for the clinicians. This results in a scenario where if a human is unsure of an answer and an algorithm displays certainty, the human uses the CNN answer. I assume this approach works because both human and algorithmic performance correlates with their perceived confidence in their answer. This approach is limited by the humans ability to correctly estimate their confidence, and I believe the resulting augmented intelligence performance will be sub-optimal because the humans will not use the CNNs answer as much as they should. In this thesis, I propose an algorithm which structures the approach. Since I do not have human confidence estimates, I invert the procedure from the described study and instead use the CNN prediction values as a proxy to determine the networks certainty. I then show that the performance improves when replacing the CNN answers with human answers on the pictures where the CNN is unsure. This methodology is fairly similar to the structure by Marchetti et al. (2020), except I look at the CNNs preditcion certainty instead of the humans prediction certainty. In this thesis, I show that this approach can improve performance with minimal cost.

The majority vote hybrid approach is a method for combining the classical majority vote approaches with Artificial Intelligence. Hekler, Utikal, Enk, Hauschild, et al. (2019) showed that adding a CNN to the ensemble works, but the study included 112 humans which is a costly affair to implement in real life. The approach needs a lot of work and would have to require much less people to be a cost-efficient solution in a real world situation. However, the idea of aggregating human and computer predictions may be a good next step in the research. I believe that combining the classical majority vote ensemble with the idea of including a CNN prediction in the ensemble can create a useful model. In this thesis, I show what happens when replacing one of the human opinions in a majority vote ensemble with a CNN prediction. I show that this can provide a low-cost method which increases prediction certainty.

1.1 Problem statement

The desire of this thesis is to develop and test the two hybrid approaches motivated above on the data set from a study by Zalaudek et al. (2006). This data set is referred to as my evaluation data set. It includes pictures and human predictions but lacks predictions from a CNN. To get these predictions, I download a training data set, build and train my own CNN, and evaluate it on the evaluation data.
Section 2 introduces the considerations regarding training data and the ways in which training and evaluation data is pre-processed for this study. Section 3 gives an in-depth overview of the building blocks of Neural Nets (NNs) and CNNs. It also explains the procedure for building networks specific to the task at hand including considerations on regularization techniques. Section 4 gives an introduction to the ROC/AUC analysis which is the primary means of evaluation in this thesis. Section 5 explains the training process and lessons learned. Section 6 reports the baseline performance of humans and CNN which lays the foundation for evaluating the hybrid performances. Section 7 introduce and evaluate the ensembling strategies. Section 8 discusses the findings along with limitations and potential applications of the hybrid methods. Section 9 concludes the thesis.

2 Data

This thesis uses two data sets. I got permission to use the data set which was used in the Zalaudek et al. (2006) study. This data set consisting of 165 pictures and corresponding human answers is good for building and testing the ensemble technique. It is referred to as my evaluation data set. However, the data set is too small to properly train a CNN, so for this purpose I use a much larger data set which I refer to as my training data set. The following sections describe the process of collecting and pre-processing the two data sets. I introduce the evaluation data first.

2.1 Evaluation data

When researching the possibilities to create hybrid approaches for skin lesion classification, I got permission to use the data from a study by Zalaudek et al. (2006). This data has to my knowledge not been used for CNN evaluation before. The purpose of the study was to show that evaluating skin lesions using a three-point checklist improved prediction accuracy and even allowed novices to obtain decent performance on the task. For this task, 165 images were randomly chosen from a collection of 2621 images. The only acceptance requirements were picture quality and haemoglobin pigmentation in all or part of the lesion. Of the 165 pictures, 15 were used for training, i.e. letting the participants get comfortable with evaluating using the three-point checklist. The remaining 150 pictures were used for evaluating performance. I use these 150 images for evaluating my CNN performance and building hybrid ensembles. The 150 images and corresponding human evaluations are referred to as my evaluation data.

The following information has been made available to me:

- JPG files in resolution 512x768 for each picture
- The three-point evaluation of each participant for each picture
- The ground truth of each picture
- Metadata on the pictures
- Metadata on the participants

Each picture in the study was presented in resolution (height = 512, width = 768) which means equates to an aspect ratio of 1:1.5. In order to train and test on similar images, the training images described in the previous section are transformed to have this same aspect ratio.

The study works with binary classfication of the images into a "bening" and a "malignant" category. The benign category is a negative class which means nothing to worry about, and the malignant category is a positive class which indicates the presence of cancer. The participants in the study were not instructed specifically to vote bening or malignant to any single picture, but instead they would rate each picture on the presence of three characteristics: Assymmetry (in colour and or structure, not shape), atypical network (pigment network with thick lines and irregular holes), and blue-white structures (blue and/or white colour within the lesion). If two or three of these characteristics are listed as being present, the lesion is classified as malignant (Zalaudek et al., 2006). In order to get the malignancy scores for each picture and participant, I convert the answers on each criteria into binary where 0 means "not present" and 1 means "present. I name these features $Guess_i$, $i = 1, 2, 3$. I then create "Guess_suspicion" which I code as

$$Guess\_suspicion \begin{cases} 0 & \Sigma_{i=1}^{3} Guess_i \leq 1 \\ 1 & \Sigma_{i=1}^{3} Guess_i > 1 \end{cases} \tag{1}$$

This gives me an answer for each participant for each picture of whether the answers indicate the image belonging to the negative or positive class.

The ground truth of each picture was evaluated through histopathological examination (Zalaudek et al., 2006). The 165 pictures contain 116 benign instances and 49 malignant instances, which is an incidence rate of 29.7 %. This incidence rate is much higher than the one in the training images which may have implications for the results. I save this discussion for section 8.

Each picture comes with metadata regarding the subject on which the photograph was taken. This metadata includes age, sex, and anatomical location of the lesion which is similar to what is provided in the ISIC data set (Rotemberg et al., 2021). However, as argued in section 2.2, I decided to not include metadata in my model.

The evaluation data included 170 participants and provided background information regarding each participant in the study. This included the participants profession, country of origin, previous experience with dermoscopy, number of yearly dermoscopies, and years of experience with the procedure. For my thesis, I include only participants with experience in dermascopy who have finished at least 126 of the 150 images. This is done to ensure my procedure is similar to a real world implementation where I imagine my results are relevant for medical professionals. The criteria of at least 126 images is to get a good estimate for the performance of each individual in the test. My criteria qualifies 69 of the 170 total participants, and my analysis later in this report is based only on those 69 participants. An overview of the relevant participant metadata can be found in table 1.

No. of test images evaluated	0	1-25	26-75	76-125	126-150	Total
Profession						
Dermatologist	16	34	10	11	54	125
General physician	3	4	2	0	6	15
Medical student	0	2	0	0	1	3
Other professional	0	4	1	1	6	12
Other medical specialty	0	2	3	0	6	11
Other	1	0	1	0	2	4
Experience in dermoscopy						
No	5	10	3	1	6	25
Yes	15	36	14	11	69	145

Table 1: Overview of profession and previous dermoscopy experience in evaluation data

2.2 Training data

Training a CNN properly takes large amounts of images. For the ImageNet Challenge, a total of 1.2 million training images were provided, divided into 1000 categories (Russakovsky et al., 2015). This means on average 1.200 images per category. For the images to be useful, they must be in decent quality and the ground

truth associated with each image must be verified. This cleaning and verification process is a rather time consuming endeavour, so it is rare to find big and useful data sets for specific computer vision tasks like the one in this thesis.
Searching for useful data sets, the first I found was the PH2 data set (Mendonça, Ferreira, Marques, Marcal, & Rozeira, 2013) which contains only two classes of skin lesions, namely melanoma and nevus. Unfortunately, the data set only contains 200 images which is far too little to properly train a CNN. Further investigation revealed that the International Skin Imaging Collaboration (ISIC) have hosted competitions in skin lesion image classification during the period 2016-2020 (*Welcome to the ISIC Challenge*, n.d.). The competition grew in size gradually through the years and in 2020 saw 3300 competitors for a price of 30,000 US dollars (*Siim-ISIC melanoma classification*, n.d.). The data sets also grew over the years. In 2016 the data set comprised only 900 images (Gutman et al., 2016). The following year, 2000 images were provided (N. C. Codella et al., 2018). For the 2018 competition, they started collecting multiple data sets into the training database. The 2018 data set comprised a total of 12,609 images including the 10,015 image HAM10K data set (N. Codella et al., 2019; Tschandl, Rosendahl, & Kittler, 2018). The 2019 data set saw a mix of some older data sets with a brand new one. It included pictures from the 2017 competition (N. C. Codella et al., 2018), the HAM10K data set introduced in the 2018 competition (Tschandl et al., 2018), and a novel data set called the BCN20000 (Combalia et al., 2019). The total data set size was 25,331 images. For the 2020 competition, a brand new data set was curated comprising 33,126 images (Rotemberg et al., 2021). A table overview of the data set sizes can be seen in table 2

Year	# of images
2016	900
2017	2000
2018	12,609
2019	25,331
2020	33,126

Table 2: Table showing the development in number of pictures used in the ISIC competition in the year 2016-2020

Reviewing the articles associated with the data sets show that a lot of preprocessing has been performed to ensure data quality. For instance, the 2020 data set is comprised of images from five different sources and only a fraction of the images passed their review process (Rotemberg et al., 2021). See table 3 for a overview of the sources and selection process for the 2020 data set.

Origin of images	images originally	images included
Hospital Clinic Barcelona	8,166	7,311
University of Queensland	26,446	8,449
Memorial Sloan Kettering Center	54,938	11,108
Melanoma Institute Australia etc.	17,616	1,884
Medical University Vienna	4,579	4,374
Finalized 2020 data set	111,745	33,126

Table 3: The origin places for the images used in the 2020 SIIM-ISIC Melanoma Classification Challenge on Kaggle.

The fact that the ISIC data sets are such high volume, purposefully curated data sets designed specifically

for the purpose of skin lesion classification, makes them an obvious choice for this project; however, I still needed to decide exactly which of the ISIC data sets of include whether to use metadata in my models, and what other pre-processing was necessary. These considerations are the focus of the following sections.

While the 2020 data set includes a large amount of pictures, it only has an incidence rate of 1.76 %. Such highly imbalanced classes may not be optimal when training an NN for two reasons. One is that training on a highly imbalanced data set may cause the algorithm to favor the majority class and thus the algorithm adapts more to the class distribution than to the image features. The second problem is that the indidence rate corresponds to only 584 positive instances which is not many pictures for a CNN to learn the features that comprise a positive case. The solution is to include the 2019 data set. This datases reveals an incidence rate of 17.85 % which corresponds to 4522 positive instances. Including this data set alleviates the issues with low numbers of positive instances, and to some degree mediates the issue with class imbalances. Reviewing the work of the 2020 Kaggle competition winners (*Siim-ISIC melanoma classification*, n.d.) shows they did similar considerations which made them decide to train on both the 2019 and the 2020 data sets. I use the 2019 and 2020 data for my thesis.

Both the 2019 and 2020 data sets included metadata regarding the patient age and sex along with the anatomic site of the lesion as. H. A. Haenssle et al. (2020) showed that having metadata (which they termed Level II information) improves diagnosis accuracy for clinicians. The reasoning is that certain types of lesions are more prevalent of may look differently depending on the metafactors. W. Li et al. (2020) have showed a similar ability in computers. They proposed a framework for image classification which fuses metadata into the classification process. Their idea is expanding on the classical CNN structure by running the metadata for each picture through a seperate NN and fuse the output activations with the extracted features from the CNN network at the flattening layer. This means the densely connected part of the CNN will be working with both the image features and the metadata features. Their research showed that the fusing of metadata yielded higher prediction accuracies compared to a standalone CNN structure without the Level II information. However, these findings are made on a different data set and with a custom built CNN which does not have a pre-trained backbone, so the findinds may not apply to my situation. Investigating the 2020 Kaggle winners revealed they built some of their models using some sort of metadata fusing algorithm (Ha et al., 2020). Those models did not show improved performance relative to the other models in their ensemble. As they did not see any performance enhancement and were able to win the Kaggle competition with mainly models that did not use the metadata, I decided to focus my time and energy on other aspects of the project.

The 2020 Kaggle competition is about binary classification of either bening or malignant, which is similar to the task in my evaluation data (see section 2.1). In the 2019 competition, the classification was multiclass and the data set was designed to accomodate this. This data sets includes a total of nine classes, of which only one is a positive instance class and eight classes belong to the negative instance. With the binary nature of the 2020 data sets and competition, it would seem natural to process the 2019 data into a binary setting. However, H. A. Haenssle et al. (2020) has shown that adding more classes in the training situation improves the model performance, with the reasoning that it gives the model more information to work with. This observation is supported by Ha et al. (2020) and would suggest that training the model for multiclass classification improves performance.
Aligning multiclass labels for the 2019 and 2020 data sets is done with inspiration from the work by Ha et al. (2020), which can be seen in table 4. Creating a binary label is done by treating the "MEL" class as the positive class and all other classes as the negative class. There seemed to be a trade-off inherent in using the multiclass labels. It could potentially yield better model performance, but as is explained in section 5, it

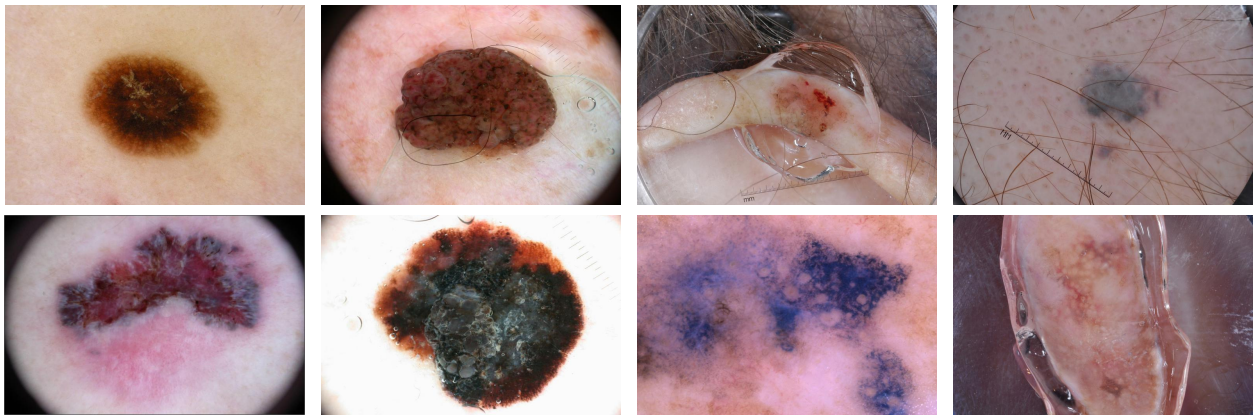

Figure 1: Top row: Four image examples from the negative categories. Bottom row: Four image examples from the positive category

comes with a risk of creating useless models. I decided the solution was to treat the modality of the label as a hyperparameter and train models on both the binary and the multiclass labels. This process is explained in depth in section 5.

2019 Diagnosis	2020 Diagnosis	Target
NV	nevus	NV
MEL	melanoma	MEL
BCC		BCC
BKL	seborrheic keratosis, licheniod karatosis, solar lentigo, lentigo NOS	BKL
AK		AK
SCC		SCC
VASC		VASC
DF		DF
	cafe-au-lait macule, atypical melanocytic proliferation, unknown	Unknown

Table 4: Table showing the alignment procedure for the 2019 and 2020 ISIC data labels. This creates the 9 target labels used in the Multiclass setting.

All images were downloaded in .jpg format in their original size. The 2019 data set was downloaded from *Welcome to the ISIC Challenge* (n.d.) and the 2020 data set from *Siim-ISIC melanoma classification* (n.d.). In order to use the trained model on my evaluation data, the image sizes must match. As explained in section 2.1 I found that an aspect ratio of 1:1.5 yielding images of resolution (512,768) would be beneficial. Inspecting the Kaggle images showed tendencies towards a similar aspect ratio, but a high degree of variance in the resolution size. I rescaled all training images into (512,768) to match the evaluation data. Figure 1 shows examples of images from the positive and negative classes from the training images.

2.2.1 Train-validation-test split

Since NNs are high capaticy and prone to overfitting, a model should not be evaluated on the same data on which it is trained (Chollet, 2017, 97). The common practice is to create three data segments: A training set, a validation set, and a test set.
The training set is the one on which the model is directly trained. These are the images which the network

with help from the optimizer tries to model in the best possible way. If the model is very high capacity, it will be able to "remember" the correct answers to each of the training instances. This is called overfitting and can be caught by comparing the performance of the model on the training set with the performance on a validation set Chollet (2017, 97-100).
The validation set is created by extracting a subset of the training set. It can be used to spot overfitting by evaluating the loss function on both the training set and the validation set after each epoch. If the training loss keeps going down but the validation loss starts increasing, this is a sign of overfitting. As such, the validation set can be used to find the point at which the models starts overfitting, thus allowing us to stop training at the right instance. Chollet warn that trying to optimize each model to the point of least validation set loss causes "overfitting to the validation set". This means that by training a large number of models, some will by chance be better adapted to the validation set than others despite not being closer to the true model. In order to mitigate this problem, a test set is introduced which is another subset of the training set. Optimizing each model for the validation set and then evaluating all models on the test set should reveal the model which is best suited for novel images.
I use a 15 % validation split to split my training data into a training and a validation set. Because the 2019 and 2020 data sets have different indidence rates, I deploy a stratified train-validation split. This means I perform both a 85-15 split on the 2019 data set and then a 85-15 split on the 2020 data set before pooling the data sets. Table 5 reveals that I achieve a fairly homogenous split in regards to the class distributions.

Category	training share %	validation share %
AK	1.5	1.6
BCC	5.6	6.0
BKL	4.9	4.6
DF	0.4	0.4
MEL	8.8	8.6
NV	31.0	30.3
SCC	1.1	1.1
Unknown	46.3	47.1
VASC	0.4	0.4

Table 5: Class distribution in training and validation sets

I do not create a test set from my training data. Instead I use the test set provided for the 2020 Kaggle competition (Rotemberg et al., 2021). This data set consists of 10,982 images posted without their ground truth. By not having the ground truth available, it is impossible to train a model specifically for performance on this test set. Instead Kaggle allows submissions of model evaluations on the test set and will score these. This allows benchmarking against the participants from the 2020 Kaggle competition.

This concludes the discussion on data selection and pre-processing. The next section outlines the mathematical foundation for evaluating model performances.

3 Neural Networks for Image classification

A Convolutional Neural Network (CNN) is a special type of Neural Network (NN) which works especially well for image classification(Chollet, 2017). This makes it an ideal algorithm type for the purpose of this thesis.

In the following paragraphs, I sum up the mechanics of NNs in general before explaining the characteristics that are believed to be the reason CNNs work so well on pictures. I then explain how pre-trained networks work and how I go about regularizing my network in this thesis.

3.1 Building blocks of NNs

Neural networks are a type of machine learning algorithm with very high capacity which can find patterns in complex data. In it's most basic form, it consists of four elements (Chollet, 2017, p. 58):

1. Layers which are combined into a network
2. Input and target data
3. A loss function
4. An optimizer

A model can have a potentially infinite amount of layers. The first layer is called the input layer, the last one is called the output layer, and all layers in between are called hidden layers. Each layer consists of a number of neurons which each contain a weight, a bias, and an activation function. The weights and biases are customarily initialized at random values and then changed at the model "learns" through training. There are many possible activation functions, but the most used used one is the rectified linear unit (ReLU) (Goodfellow, Bengio, & Courville, 2016; Sharma, Sharma, & Athaiya, 2017). A ReLU activation can be expressed mathematically as $f(x) = max\{0, x\}$ (Goodfellow et al., 2016, p. 187) which means $f(x|x < 0) = 0$ while $f(x|x > 0) = x$. This ensures that the network works with linear, non-negative activations. While many other types of activation functions exist and could be used, the ReLU has proven to perform strongly across a broad range of scenarios. In addition, it is computationally "light" and allows much faster training and convergence compared to many other activations (Rawat & Wang, 2017).
For the output layer, the ReLU is not the obvious choice. The appropriate structure for this layer is highly task specific. For regression problems, eliminating negative outputs with a ReLU is a major mistake. Instead, a linear activation $f(x) = x$ is commonly used (Chollet, 2017). When doing binary classification, Chollet (2017, p. 320) recommends always using a sigmoid activation which can be described as $f(x) = \frac{1}{\exp^{-x}}$ (Sharma et al., 2017). The sigmoid activation compresses the outcomes into a $[0, 1]$ range and is similar to a logistic regression function. For binary classification ($K = 2$), only one neuron is used in the output layer which means only one value is between zero and one is output from the network. A typical next step for classifying is to set a threshold t and introduce so that

$$(x) \begin{cases} 0 & f(x) < t \\ 1 & f(x) \geq t \end{cases} \tag{2}$$

Where $g(x)$ is the classification decision and f(x) is the prediction probability from the model. For the multiclass setting with ($K > 2$) categories, Chollet (2017, p. 320) recommends using a "softmax" activation. In this context, the number of neurons in the output layers should equal the number K of classes in the data. The softmax function is a combination of multiple sigmoid functions and can be expressed as (Sharma et al., 2017, p. 314)

$$f(x)_j = \frac{\exp^{x_j}}{\Sigma_{k=1}^{K} \exp^{x_k}} \tag{3}$$

for $j = 1, \ldots, K$ where j represents the j'th of the K classes. For this thesis, a sigmoid activation is used for the binary target situations and a softmax is used for the multiclass target situations.

When training a model, the input data is fed through the model and the output is compared to the target data using a loss function. If the model guesses correctly, the loss is low and vice versa. For regression problems, the most common loss functions are Mean Squared Error (MSE) or Mean Average Error (MAE). The MSE can be expressed as (Goodfellow et al., 2016, p. 105)

$$MSE = \frac{1}{m}\Sigma_i(\hat{y}^{(test)} - y^{(test)})_i^2$$

where $\hat{y}^{(test)}$ are the model predictions for each test point and $y^{(test)}$ are the actual target values. m represents the number of observations. This score is essentially an averaged Eucledian distance function which yields high values if the predictions and targets are highly dissimilar, and yields low values if the predictions are close to the targets.
MAE is an averaged distance measure like MSE, but it does not squared the distances between the individual predictions and the targets (Chollet, 2017, p. 212).

$$MAE = \frac{1}{m}\Sigma_i \hat{y}_i^{(test)} - y_i^{(test)} \tag{4}$$

For classification settings, the output is not linear and it is restricted to the $[0, 1]$ interval. For this reason, it is more appropriate to use crossentropy as the loss function. The derivation for the cross-entropy formula in Goodfellow et al. (2016, p. 71-73) is rather advanced. It is essentially a quantity highly related to the Kullback-Liebler divergence and takes on the formula

$$(P, Q) = -\mathbb{E}_{x \sim P} log Q(x) \tag{5}$$

Minimizing the cross-entropy corresponds to maximizing the likelihood function (Zaki & Meira, 2020, p. 626). This means that by minimizing the cross-entropy, we maximize the likelihood that the model output is correct. This project works with categorical data and uses cross entropy as the loss function for all models.

The purpose of the optimizer is to minimize or maximize the loss function (Goodfellow et al., 2016, p. 79). For the cross entropy loss used in this project, the goal is to minimize the loss function. To do this, we must first find the gradient descent. Suppose the loss function is expressed as $y = f(x)$ where both x and y are real numbers. The derivative of this function, $f'(x)$, tells us the slope of $f(x)$ at the point x. By moving small steps in the opposite direction of the slope, we reduce $f(x)$. When $\boldsymbol{x}$ is a large input space such as the weights and biases of a neural network, we take the partial derivative of each input $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ and create a vector containing all the partial derivatives called $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. We can now define the method of gradient descent as

$$\boldsymbol{x'} = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x} \tag{6}$$

Where ϵ is the learning rate. By moving ϵ-sized steps along the gradient, the loss function eventually converges towards a minimum where every element in the gradient is close to zero (Goodfellow et al., 2016, p. 80-82). We can never be certain if this minimum is a local or global minimum, so contemporary optimizers have been designed advanced techniques like with momentum and adaptive learning rates which help them not get stuck in local minima. This thesis uses an optimizer named Adam which uses stochastic gradient descent based on first- and second-order moments (Kingma & Ba, 2014) to help it find a global minimum.

3.2 Characteristics of CNNs

The described network structure is the general form of a Neural Network. A basic, fully-connected NN can have extremely high capacity, but is limited by the fact that it can only handle a flat input space. This is alleviated with CNNs which are a special type of NNs that use hidden layers called Convolutional layers and Pooling layers. These layers allow the network to take as input a 3D Tensor with input dimensions (height, width, color) which means the relative position of the pixels remain intact (Chollet, 2017). The Convolutional layers search the input picture in a grid like structure, i.e. 2x2 pixels at a time. This allows the network to learn patterns which are translation invariant (Chollet, 2017, p. 123). The pooling layers also work in grid structures and downscale the image by aggregating nearby pixels. In combination, the convolutional and pooling layers allows the network to learn spatial hierarchies of patterns Chollet (2017, p. 123). This process of decoding pictures inductively from local to global patterns seems intuitively similar to the way humans process images.

In a CNN, the output from the convolutional- and pooling layers contain the generalized image patterns. In order to translate this into an prediction, the onvolutional output must be fed through some type of flattening layer followed by some densely connected layers that end in an output layer. I shall refer to the convolutional part of a CNN as the base or backbone, and to the densely connected part containing the output as the head of the network. Developing and training good CNNs is a complex process which requires large amounts of data and vast amounts of computer power. The gradient descend calculations associated with NNs combined with the ability to have millions of trainable parameters makes for a computationally difficult task. For this reason, it is common in image recognition tasks to build upon pre-trained models which have previously been trained for general image recognition and can then be relatively easy specialized for specific problems (Rawat & Wang, 2017). The next section will introduce and explain the fundamentals of pre-trained models and how to utilize them.

3.3 Pre-trained models

When Alex won the 2012 ImageNet Challenge, it did so with a top-5 accuracy of 84.7 % (S. Han et al., 2018). This means that when presented an image which could represent one of 10,000 categories, the model would accurately classify the image 84.7 % of the times if given five guesses. The next years saw rapid improvement in the capacities and performance of the models, and in 2017 almost all participants in the competition had achieved top-5 performances of $> 95\%$.
Due to the open-source culture of computer science, many of the high performance CNN models have been made publicly available. These models, which are called pre-trained networks, are of great importance to this project as they have been proven to be easily trainable on novel image recognition tasks (Rawat & Wang, 2017). During the literature review, all but one articles found dealing with skin lesion classification used one or more pre-trained networks as a backbone for their model. An overview of the studies and the pre-trained models used can be found in table 6.

The idea behind using ImageNet Challenge winners is that these networks have a well trained base which has effectively learned the spatial hierarchies of features (Chollet, 2017). The base thus acts as a generic models of the visual world. By removing the head of such model and replacing it with a new head, it is possible to specialize the network in new recognition tasks much faster than by training a new CNN from the bottom-up. Adapting the network to a new setting can be done in two ways, feature extraction or fine-tuning (Chollet, 2017). In feature extraction, the base of the model is frozen meaning the parameters are not trainable. The head of the model is replaced with a new head which is randomly initialized. Only the

Author	Network(s) used
S. Han et al. (2018)	ResNet-152
T. Brinker et al. (2019)	ResNet50
Esteva et al. (2017)	Inception V3
Tschandl et al. (2020)	ResNet34
Hekler, Utikal, Enk, Hauschild, et al. (2019)	ResNet50
Mahbod et al. (2018)	AlexNet, VGG16, VGG19, GoogleNet, ResNet-50, ResNet-101, ResNet-152, Inception-V3, Inception-V4, DenseNets, SeNets, PolyNets
H. Haenssle et al. (2018)	InceptionV4
Tschandl et al. (2019)	Inception-V3, ResNet-50
Hekler, Utikal, Enk, Berking, et al. (2019)	ResNet50
T. J. Brinker et al. (2019)	ResNet50
H. A. Haenssle et al. (2020)	MoleAnalyzerPro*
Ha et al. (2020)	EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, EfficientNetB7, Se-ResNext101, ResNest101
S. S. Han et al. (2020)	SeNet, Se-ResNet50, VGG19
W. Li et al. (2020)	Custom**

Table 6: All except one reviewed articles dealing with skin lesions classification has used one or more pre-trained networks.
* MoleanalyzerPro is a medically approved network developed in Germany.
** This article did not attempt to build a state-of-the-art network, but rather to show through a custom setup that fusing metadata into the decision process improved performance.

head is trainable and this is now trained by feeding images through the full network and using an optimizer to teach the head how to respond to the training images. In fine-tuning, the process is fairly similar to feature extraction, but it works by un-freezing parts of the base layers. This is a delicate process which requires more computational power but is able to achieve higher predictive power. It can easily wrong if strong gradients back propagate through the base layers and disrupt the abilities of these, so Chollet (2017, 154) recommends doing fine-tuning using the following process:

1. Import a pre-trained network without the head and add a custom head on top of it.
2. Freeze the base of the network, making it unable to train.
3. Train the network.
4. Unfreeze part or all of the base network.
5. Train the network again.

That structure for fine-tuning works well as a guideline, but the "no free lunch" theorem holds true for this project (Ho & Pepyne, 2001) and as I explain in section 5, many adjustments are made before the final training setup.

3.3.1 Choosing a pre-trained model

With more than 30 pre-trained networks to choose from (Team, n.d.), considerations were done. As can be seen in table 6, ResNet networks are very popular backbones for skin lesion analysis. However, the winning submission from the 2020 ISIC Kaggle Challenge used primarily EfficientNets (Ha et al., 2020). An article by Tan and Le (2019) describes how the EfficientNets generally outperform other networks at similar numbers of parameters. This could indicate the networks are relatively cost-effective as less parameters require less computational effort to train the network. An overview of the pre-trained networks in the Keras package confirms the efficiency of the EffientNets (Team, n.d.). They are achieving very solid performance numbers on the ImageNet Challenge while having relatively few parameters and reasonable training times. With limited computational resources, choosing an EfficientNet is a logical choice for this study. I decided to use the EfficientNet B3 variation which is the simplest of the models used by (Ha et al., 2020). An overview of the base of this network can be found in the Appendix section 11.1, which is copied from (Alhichri, Alsuwayed, Bazi, Ammour, & Alajlan, 2021).

3.4 Regularization

Deep Neural Nets including Deep CNNs are high capacity models. This makes them prone to overfitting, which means high performance on a training set but poor performance on a held-out testing set (James, Witten, Hastie, & Tibshirani, 2013; Rawat & Wang, 2017). The methods used to combat overfitting are called reguralization techniques and there are many of them. Experimenting with regularization techniques and finding the optimal combination (measured using a held-out validation set) is called hyperparameter tuning. For the final training in this project, I use and optimize on four techniques that are prevalent in the reviewed literature Chollet (2017); Goodfellow et al. (2016); Rawat and Wang (2017):

- Data augmentation
- Dropout layers
- Capacity regulation
- Weight reguralization

The recommendation by Chollet (2017) is to experiment with model capacity until the point where the capacity is a little too high and signs of overfitting start showing. At this point, one should start dialing the model back using the above regularization techniques until a good test set performance is achieved. There is no single way to do this (Chollet, 2017; Goodfellow et al., 2016), but rather it is an iterative process which can take a long time and draws highly on the experience of the data scientist who is developing the algorithm.

The following sections will describe the above mentioned regularization techniques and how they apply to this project. The discussion of how the parameters are combined and evaluated is provided in section 5.

3.4.1 Data augmentation

Data augmentation means altering the data before presenting it to the network. A theory of overfitting is that is occurs when the network looks at the same image too many times and learns to recognize the exact pictures rather than then features of the picture (Chollet, 2017). This is especially a problem with smaller data sets where data augmentation can combat the small data variation by creating new images from the existing ones by rotating, zooming, shearing etc. the existing pictures. For larger data sets, benefits can also

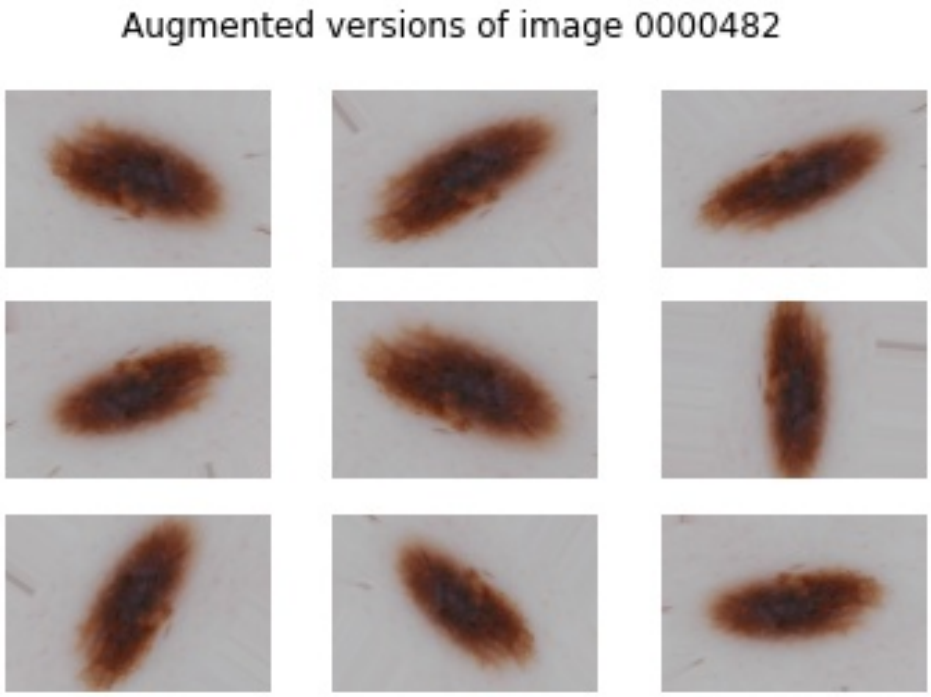

Figure 2: 9 augmented versions of the same picture. The augmentation techniques are zoom, rotation, shear, and horizontal flip. The picture is from the actual data set used for training and is an example of a malignant skin lesion

be gained using data augmentation as it can greatly improve the models generalization ability (Goodfellow et al., 2016, 233). I use zoom, horizontal flipping, and rotation as data augmentation strategies for this project. This process is shown in figure 2 which shows 9 versions of the same picture, augmented in different combinations. To a human, it is clear that the pictures include the same mole; however, to a machine they are nine different moles and will help the model generalize to novel images where the moles are oriented in novel ways.

I also considered using shear which is a popular augmentation strategy for other computer vision tasks, but this influences symmetry perception and Zalaudek et al. (2006) showed that asymmetry is a highly predictive feature for malignant melanomas. As such, including shear may make asymmetric images symmetric and thus cheat the network of useful information.
I believe data augmentation is such an essential reguralization technique that I use it in all my models. This is based on my own experience building computer vision models along with my observations that every reviewed article included data augmentation in their final model.

3.4.2 Dropout layers

Dropout provides an inexpensive method of approximating a bagged ensemble of many neural networks (Goodfellow et al., 2016, 251). Consider dropout layer D which lies between two hidden layers H_n and H_{n+1} in a fully-connected part of a Neural Network. The dropout layer works by initiating a binary mask

which is multiplied onto the output form H_n before inputting the activation to layer H_{n+1}. This effectively renders part of the activation inactive and forces layer H_{n+1} to work with less than full information. Since the binary mask is initiated randomly at each minibatch, the dropout layer turns the model into a bagged ensemble of highly correlated models without having to instantiate multiple models. I use dropout in my model by setting up a list with four different dropout values (0, 0.2, 0.4, 0.6). This is done with inspiration from an article by Tan and Le (2019) who explained how they used different dropout values when training their different EfficientNet models, as they found models with high capacity should be penalized with higher dropout values.

3.4.3 Capacity regularization

The capacity of a NN can be regulated by adjusting the number of layers (the depth) and the number of neurons in each layer (the width) of the network. Increasing depth or width increases the number of weights and biases, meaning more parameters in the network. Finding a good network structure and a good balance between the width and dept of a network was the purpose of the research which ended in the development of the EfficientNets (Tan & Le, 2019). When building and optimizing a network, Chollet (2017, 114-115) recommends building the network too complex up to the point where overfitting is evident and then deploy general regularization techniques. This approach is not viable for my thesis as the data set and models are highly complex which means training even a simple model takes multiple days. Instead I create multiple models with different capacity and different reguralization strategies and train simultaneously with the hope of finding a descent solution. My different networks are described in section 5.

3.4.4 Weight decay

Weight decay is a reguralization technique that imposes a "cost" function which penalizes complexity. This directly decreases the complexity and thus ability of the network to overfit. ℓ^2 weight decay, also called "ridge regression" is the most common form of weight decay (Goodfellow et al., 2016). Assume the objective function is expressed as

$$J(w; X, y) \tag{7}$$

The reguralized function can then be expressed as

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^\top w + J(w; X, y) \tag{8}$$

where α is the weight parameter determining the relative contribution of the norm penalty function. w represents the weights that can be affected by the weight decay. If $\alpha = 0$, there is no regularization and $\tilde{J} = J$. If $\alpha > 0$, the model will be punished for having large weights and thereby being highly complex. A good α value is situation specific and should be found through experimentation. I initially experimented with using ℓ^2 as a kernel regularizer for my models with the values [0,0.01,0.001]; however, implementing it into my network with a pre-trained base caused some errors in TensorFlow and eventually caused the full model to crash repeatedly. I decided my time was better spent working on the remaining parts of my project than trying to work around the code conflicts.

This concludes the theoretical explanation of CNNs and how they relate to my thesis project. The next section introduces the principle behind ROC curves which are used to evaluate my models and lay the foundation for building my ensemble algorithms.

4 ROC curves

When evaluating a model performance on a classification task, a common and easy-to-use measure is the prediction accuracy. This measure is defined as (Zaki & Meira, 2020, 547):

$$Accuracy = \frac{1}{n}\Sigma_{i=1}^{n} I(y_i = \hat{y}_i) \tag{9}$$

Where n = number of observations, y_i is the correct answer, $\hat{y}_i$ is the guess by the model and I is an indicator function that takes the value 1 when its argument is true. In essence, accuracy is calculated by comparing each guess with the correct answer and then dividing the total number of correct answers with the total number of answers. However, the accuracy measure may not be a good solution for imbalanced data sets (Zaki & Meira, 2020), since a no information model predicting only the majority class can achieve high accuracy and still be useless. In these cases it can be useful to perform a Receiver Operating Characteristic (ROC) analysis because it is insensitive to class imbalances (Zaki & Meira, 2020, 560). In addition, the relation between the floating point outputs from a CNN and the thresholds used in a ROC analysis makes this analysis type very popular among the literature on CNN evaluation of skin lesions. I elaborate on this in the next section, after introducing the basics of the ROC analysis.

The ROC analysis builds on the concepts of True Positive Rate (TPR) and False Positive Rate (FPR) which I explain using confusion matrices. Consider a binary classification setting where we attempt to classify n observations. Let c_1 represent the positive class, c_2 the negative class, and let y_i and $\hat{y}_i$, $i\{1, 2 \ldots, n\}$ represent the true and predicted label label for each observation. We can now create a confusion matrix which includes the following four scenarios (Zaki & Meira, 2020, 551):

- True Positives (TP): The number of observations that the classifier correctly identifies as positives: $TP = |(\mathbf{x}_i|\hat{y}_i = y_i = c_1)|$
- True Negatives (TN): The number of observations that the classifier correctly identifies as negatives: $TN = |(\mathbf{x}_i|\hat{y}_i = y_i = c_2)|$
- False Positives (FP): The number of observations the classifier predicts to belong to the positive class, but which actually belong to the negative class: $FP = |(\mathbf{x}_i|\hat{y}_i = c_1 \wedge y_i = c_2)|$
- False Negatives (FN): The number of observations the classifier predicts to belong to the negative class, but which actually belong to the positive class: $TP = |(\mathbf{x}_i|\hat{y}_i = c_2 \wedge y_i = c_1)|$

By counting the number of observations which fall into each scenario, we can calculate the True positive rate (TPR) and the False positive rate (FPR). The TPR is the ratio of observations classified as positive that are actually positive and the FPR is the ratio of observations that are classified as positive but are actually negative:

$$TPR = \frac{TP}{TP + FN} \tag{10}$$

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

If we express accuracy using a confusion matrix, we realize how closely related TPR and FPR are to that measure (Zaki & Meira, 2020, 552)

$$Accuracy = \frac{TP + TN}{n} \tag{12}$$

But whereas the accuracy score is concerned with both classes, the ROC analysis is only interested in the performance on the positive class.

When performing a ROC analysis, a plot is typically made showing the FPR on the x-axis and the TPR on the y-axis (Zaki & Meira, 2020, 558). This is highly used in the literature to show and compare the performance of individuals and groups to each other and to ML algorithms (T. Brinker et al., 2019; T. J. Brinker et al., 2019; Esteva et al., 2017; H. Haenssle et al., 2018; H. A. Haenssle et al., 2020; S. Han et al., 2018; S. S. Han et al., 2020; Hekler, Utikal, Enk, Hauschild, et al., 2019; R. Kurvers et al., 2021, 2015; W. Li et al., 2020; Marchetti et al., 2020; Tschandl et al., 2019). When showing human performance, each participant or group is typically expressed as a single point in the plot representing the (FPR,TPR) values. But when evaluating the performance of an ML algorithms such as a CNN, it is possible to draw a line of corresponding (FPR,TPR) values. This line is made possible with the probability estimates that are generated by many machine learning models, i.e. CNNs. Imagine sliding a certainty threshold t in range $[0, 1]$ and for each value of t convert each prediction value $p_i, i \in 1, 2, \ldots, n$ in the prediction array into binary labelled predictions $pred_i, i \in 1, 2, \ldots, n$ using formula 13.

$$pred_i \begin{cases} 0, p_i < t \\ 1, p_i > t \end{cases} \tag{13}$$

For each prediction array, we can record the corresponding (FPR,TPR) values and show the relationship between FPR and TPR at different thresholds. The algorithm for this can be seen in algorithm 1. Once the graph is drawn, an Area Under the Curve (AUC) can be calculated as the integral of the curve. Random guessing will yield an AUC score of 0.5, and perfect distinction between the binary classification categories will yield an AUC score of 1 (Zaki & Meira, 2020). A common approach in the reviewed literature is to plot the ROC-curve for the ML algorithm and the (FPR,TPR) value pairs for the humans or groups of humans in the same graph. If the curve the for ML algorithm lies "above and to the left" of the human performance, the intuition is that the ML has a stronger performance than the humans because the ML algorithm has higher TPR than humans at a similar FPR and vice versa lower FPR than humans at a similar TPR.

This study performs the ROC/ AUC analysis in a few different ways for the various data sets. On the training and validation data set, the analysis is performed using the built-in AUC measure in the Tensorflow Keras package. The package uses as default 200 threshold values to perform the analysis and I see no reason to customize this number. Although ROC/AUC analysis is originally designed for binary classification, this package has an option of performing multiclass AUC analysis by flattening the data into a single label before computing the AUC (*Tf.keras.metrics.AUC; Tensorflow core v2.8.0*, n.d.). For the test set, the AUC score is calculated on Kaggle using the "Late Submission" option for their 2020 competition (*Siim-ISIC melanoma classification*, n.d.). This requires submitting a dataframe containing the ID of each test photo along with the predicted probability for the "melanoma" class. For the evaluation data, two different approaches are used for the human and computer performances. The human performance is evaluated by making a confusion matrix using the SciKit-Learn (SK) module and extracting the TPF and FPR from this matrix. The CNN performance is evaluated by performing a ROC/AUC analysis using another SK module. This method extracts the (FPR,TPR) pairs for up to 200 thresholds and gives an estimate of the AUC score. The pairs are plotted using the pyplot extension of the matplotlib package.

This concludes the introduction to ROC analysis. The next section introduces my procedure for training my CNN along with the lessons learned and adaptations made along the way.

Algorithm 1: ROC/AUC Analysis

input : Predictions, Ground truth observations, thresholds

output: ROC Curve, AUC Score

```
TPR = list
FPR = list
foreach value in threshold do
    t ← threshold
    TP ← 0
    FP ← 0
    TN ← 0
    FN ← 0
    foreach image in Evaluation images do
        p ← probability of malignancy
        truth ← ground truth observation
        if p > t then
            prediction ← 1
        else
            prediction ← 0
        if prediction = 1 & truth = 1 then
            TP ← TP + 1
        else if prediction = 0 & truth = 0 then
            TN ← TN + 1
        else if prediction = 0 & truth = 1 then
            FN ← FN + 1
        else
            FP ← FP + 1
    end
    Update TPR ← add TP/(TP+FN)
    Update FPR ← add FP/(FP+TN)
end
Plot = Lineplot showing FPR and TPR pairs
AUC = Integral of plot
Return Plot, AUC
```

5 Training a CNN

When it comes to building a high performing Neural Network, there are no formal rules (Chollet, 2017, 263). Experienced data scientists achieve an intuition for what techniques may apply to certain situations, but even so the initial setup is almost always sub optimal. As described in section 3.3, using a pre-trained network from the ImageNet Challenge as backbone is the common way to approach my task of training a CNN for skin lesion classification. This approach has been used in numerous research articles (see table 6), it has been used to win a large Kaggle competition (Ha et al., 2020), and it is recommended in books dedicated to Deep Learning (Chollet, 2017; Goodfellow et al., 2016).

Part of the computation done for this project was performed on the UCloud interactive HPC system, which is managed by the eScience Center at the University of Southern Denmark. Specifically, all the full training of models was performed on a node with an Intel(R) Xeon(R) Gold 6130 CPU@2.10 GHz, 32 vCPUs, and 192 GB of memory. The code is developed in Python 3 and is tested to work with Tensorflow versions 2.6.0 and 2.8.0

The next sections describe my attempts to build and train a CNN. First is a description of my initial attempt and why I had to scrap this approach and follow a different path. Next is an explanation of the research and observations which guided me toward a succesful network. The final section is a description of the final training setup.

5.1 Initial training attempt

My initial desire was to train a CNN using the EffiCientNetB3 (Alhichri et al., 2021) with a Flattening and three Dense layers on top. I wanted to follow the five step checklist by Chollet (2017, 154) which is described in section 3.3 and build my model structure similar to the one described by Chollet. The desire was to explore the following hyperparameter search space (potential values in parenthesis):

- Adding a Dropout layer after the Flatten layer (True, False)
- Size of the first Dense layer (range [64,512], step = 64)
- ℓ_2 regularizer for the first layer (0.0, 0.1)
- Adding a batch normalization layer after the first Dense layer (True, False)
- Size of the second Dense layer (range [32,256], step = 32)
- ℓ_2 regularizer for the second layer (0.0, 0.1)
- Learning rate for the optimizer (range [0.01, 0.0001], sampling = logarithmic)

This parameter space is substantially larger than the one described in section 3.4. An exhaustive search of it (excluding the learning rate) would require training 1024 models. I knew this was not feasible so I wanted to implement a Keras tuner package named Hyperband (L. Li, Jamieson, DeSalvo, Rostamizadeh, & Talwalkar, 2017). This is a multi-armed bandit approach which iteratively builds models within the hyperparameter space and keeps track of their performance. The tuner takes a greedy approach of building a subset of models from the parameter space, trains them for a few epochs, saves each model and then evaluates the performance of them. The worst performers are discarded and the top performers are loaded from their trained state. The process repeats where the top performers are trained for some more epochs before being evaluated again. In this way, little energy is wasted on training poorly performing models and more energy is used on training and finding the top performers.

My initial test using the Keras tuner ran on my own computer. It was performed on 320 training images ($< 1\%$ of the total 50K) and 160 validation images in size (192,158). The maximum number of epochs was 15 and the total runtime was ≈ 7 hours. The Hyperband tuner ran 90 models and saved the best model. Afterwards I unfroze the base of this best model and trained it for an additional 10 epochs with a learning rate of $1e-5$ as per the procedure described in section 3.3. The model summary in table 7 shows a model with more parameters in the first dense layer than in the base. The large numbers of parameters is due to the large output size from the flattening layer combined with the large numbers of parameters in the first dense layers.

Layer	*Output shape*	*Number of parameters*
efficientnetb3 (functional)	(None,6,5,1536)	10783535
flatten (Flatten)	(None, 46080)	0
dropout (Dropout)	(None,46080),	0
Dense1 (Dense)	(None, 256)	11796736
batch_normalization (BatchNorm)	(None,256)	1024
Dense2 (Dense)	(None,160)	41120
dense (Dense)	(None, 8)	1288

Table 7: Model summary of the best model from my initial low-scale attempts at parameter tuning

Evaluating the results from this initial model revealed four severe issues. First I realized that the picture size was completely off. I had confused the width and height dimensions and I had messed up the aspect ratio. Instead of (192,158), I should have encoded the images as (128,192).
The second issue was the potential runtime of running the test full scale. Testing on the cloud computer revealed the 32 nodes were used suboptimally with constant spikes and valleys in the performance output during training. Training times per epoch with this configuration were close to those achieved on my own computer. Attempts with upscaling images to full size (512,768) increased the time per epoch with a factor of ≈ 12. An estimate for the runtime of a full model (FM) based on this initial model (IM) would be

$$FM = IM * upscale\_factor * \frac{FM_n}{IM_n} = 7 * 12 * \frac{49689}{320} = 13043 \tag{14}$$

where FM_n is the number of pictures in the full traning set and IM_n is the number of pictures in the initial training set. This estimate is not entirely correct as it does not take into account the change in validation set size, but it is sufficient to see that a full training run is infeasible.
The third issue was training data. I thought I had downloaded all relevant training data as described in section 2.2 but I had only downloaded the HAM10K part of the 2019 data set. Because this data set had very few observations in some classes, I had chosen to encode only eight classes instead of nine. Once I downloaded the full data set I re-encoded it to fit with the description in section 2.2.
The last issue with my training run came when I evaluated it. Despite achieving a low loss value, the model did not seem to have learned anything useful about the data set. Inspecting the predicted probabilities revealed the model had learned to give high probabilities for the prevalent classes and low probabilities for the uncommon classes. Table 8 shows some descriptive statistics on the probabilities from this initial model. It is clear in the table that the model prefers category 6 and in general has very little variation in its prediction probabilities.

I realized my initial approach was not a viable solution and took a step back to meet with a counsellor and review how others had approached this specific problem.

5.2 Restructuring

My initial approach proved futile and with no immediate resolution in sight there was a potential for wasting much time and effort before having a decent model. My counsellor suggested looking for a publicly available model trained specifically for the Kaggle competition. Researching the winning solution to the 2020 Kaggle competition (Ha et al., 2020; *Siim-ISIC melanoma classification*, n.d.) revealed their code was publicly available on Github (Haqishen, n.d.). Their script had trained each of the 18 ensemble models for 15-45 hours on a setup with eight NVIDIA Tesla V100 GPUs which was far above my available resources.Their

	Category							
Label	0	1	2	3	4	5	6	7
Min	4.5e-17	1.3e-14	2.5e-13	1.8e-18	2.8e-12	0.012	0.85	1.7e-18
Median	5.9e-10	1.1e-8	9.0e-8	9.8e-11	4.5e-7	0.085	0.92	1.3e-10
Max	4.3e-5	1.5e-4	4.5e-4	1.9e-5	5.7e-4	0.15	0.99	1.4e-5

Table 8: Descriptive statistics of the probability estimates from the initial training round. Low loss functions were achieved, but the accuracy and AUC scores were low as well. This could indicate the model has optimized the loss by predicting the dominant class in the training data rather than learn to distinguish the classes

model weights were also published (Bo, 2020), but their models were built for the PyTorch package all my work was built around Tensorflow/Keras. I did not manage to build a network and implement their weights. Additional research on the Kaggle competition (*Siim-ISIC melanoma classification*, n.d.) revealed competitors who had posted their code for inspiration to others. A Kaggle grandmaster named Chris Deotte (Cdeotte, 2020) had published his code to train his model which got him a 53^{rd} place finish in the competition of more than 3000 teams. This script was build to run on TPU which was not an option for me. The weights were published, but not the full model. Also, the model was adapted for a 1:1 aspect ratio which would force me to rescale my evaluation pictures. Another submission to the competition from Jang (2020) had also posted some code for the competition. Her submission score was not made public, but the simplicity of her script and the markedly shorter training times compared to other scripts lead me to believe it was not a top performer. The script was also made for training on TPU so it was not an option. But she had published her full model which I downloaded and evaluated on my evaluation pictures which forced a rescaling into (256,256). The model performed with an AUC score of 0.773. I include a visualization of this ROC curve in Figure 5 to compare with my own baseline model.

I did many observations during the review and tested them out on a small scale on my own computer. The alterations gave rise to models which actually seemed to learn from the pictures rather than the class distributions. These observations were:

- Using a GlobalAveragePooling layer. Both Deotte and Jang used a GlobalAveragePooling layer instead of a Flattening layer to connect the base and head. This alteration had two major advantages. For one, it radically improved model performance. Second, it drastically deceases the amount of parameters in the first dense layer of the head. Running a (512,768) resolution picture through the EffiCientNetB3 network produces an output of shape (16,24,1536). Running this through a flattening layer gives an output of size $16*24*1536 = 589824$. If the first Dense layers has 512 neurons, this results in more than 300 million parameters in that layer. Trying to train this model would be highly unfeasible. Running the base output through a GlobalPoolingLayer produces an output of size 1536. This brings the same Dense layer with 512 neurons down to 786944 parameters which is much more feasible.
- Training with a binary labelled training set. I noticed that Jang (2020) had encoded her training set with binary labels. This poses a trade-off between model information and data complexity. A binary label gives the model less information to "learn" which is theorized to be inferior to multilabel training (Ha et al., 2020). But with my initial results suggesting the model learns only the class distribution, perhaps the image classification is too complex for my models. I tried testing with binary labels and quickly got decent results.

- Using a learning rate schedule. Chollet (2017) suggests doing fine-tuning by using a learning rate in range $[1e-4, 1e-2]$ for the initial training of the head, then unfreezing some of the base layers and lowering the learning rate before training the model further. None of the models reviewed seemed to follow this approach. Instead they unfroze the full model and implemented learning rate schedules with very low learning rates. Jang (2020) used an exponentially decaying learning rate starting at 0.01. Cdeotte (2020) and Ha et al. (2020) used a slightly more complicated learning rate schedule. Their schedules ran the first epoch as a "warm up" epoch with extremely low learning rate. The learning rate would then spike to a maximum for the second epoch before decaying slowly again. Cdeotte (2020) calls this "a common train schedule for transfer learning". The algorithm for my learning rate schedule can be seen in Appendix section 11.3
- Creating models with shallow heads. The model by Jang (2020) had only two fully connected layers - one with 8 neurons and the output layer with 1 layer (binary classification). The model by Cdeotte (2020) only had the output layer with 8 neurons (multiclass classification). The two models have radically less parameters than what Chollet (2017) proposed.
- Using class weights. Jang (2020) introduced class weights to the optimizer in her model. Since the classes are highly skewed, this approach makes sense and should counteract the problem with models predicting the category distribution. The weight w_i for each of the $i\{1, 2, .., k\}$ classes can be calculated as:

  $$w_i = \frac{n_{tot}}{n_i * k}$$

  Where n_{tot} represents the total number of observations in the training set and n_i represents the number of observations in the training set belonging to class i.

Testing the above mentioned methods showed radically improved performance of my models. Model predictions now revealed a tendency for my models to actually learn features regarding the pictures rather than just the distribution. I decided to combine the approaches and build a new model with the intention of beating the evaluation set performance of the model by Jang.

5.3 Final setup

For my final setup, I had three parameters in my grid: Label encoding (binary/multiclass), model head capacity (shallow/deep), and strength of dropout layer (0/0.2/0.4/0.6). I initially wanted to only have two sizes of dropout layers (0/0.5) and then incorporate an ℓ_2 kernel reguralizer in the first Dense layer as described in section 3.4. However, this did not implement well with the EfficientNetB3 structure and I kept getting error messages. Since none of the reviewed models seemed to have implemented this reguralizer in their final models, I decided it was not worth trying to implement. Instead I expanded the droput layer options. The grid comprises a total of 16 models.

To ensure the models could train long enough to converge but no longer than necessary to reach conversion, I set the maximum number of epochs to 50 and implemented an early stopping callback with patience 7. This implementation saves the best model as measured on the loss value for the validation set. Once a best model has been found, if this does not improve over the next seven epochs, the training terminates. The early stopping callback can save vast amounts of time and computational resources.
Inspecting a guide for using the pre-trained EfficientNet models (Yixing, 2020) showed the base model is

designed for pictures with resolution 300. It can easily work with higher resolutions, but this gives higher computational demands and no guarantee of a better model performance. Downscaling my images with a factor of two results in a resolution of (256, 384) which seems like a reasonably choice. I scaled down all training, validation, testing and evaluation images to this size.
Preparatory trials revealed a training time of just over two hours per epoch. Assuming 15 epochs of training per model like (Ha et al., 2020) used, this would take 35-40 hours including evaluation per model. Running this sequentially could take too much time for this project, so I decided to split up the models and train them in parallel. With the nature of the hyperparameter grid, it made sense to split up the training into four segments based on target label and model capacity. Four scripts were built with each a unique model structure, each running the four dropout rates in a grid.

- A model with a "deep" head consisting of four Dense layers with 128, 32, 8 and 1 neuron. This model was used for binary classification.
- A model with a "shallow" head consisting of two Dense layers with 8 and 1 neuron. This model was used for binary classification.
- A model with a "deep" head consisting of three Dense layers with 128, 32, and 9 neurons. This was used for multiclass classification.
- A model with a "shallow" head consisting of on Dense layer with 9 neurons. This was used for multiclass classification.

All scripts include the same data pre-processing. This includes an ImageDataGenerator which scales all images to size (256,384) and augments the training images as described in section 2.2. The ImageDataGenerator feeds the images from memory into the network in randomly selected batches of a certain size. If the batch size is too small, many of the batches will not include the marginal categories in the distribution, so it is important to set the batch size as large as possible. The upper limit for the batch size is given by the cache size of the CPU. Too much information at the same time will overload the processor and either slow things down substantially or cause the script to terminate. I use a batch size of 64 for the training and validation data. For the test data, I use a batch size of 1 and disable the random selection procedure to ensure I can get the pictures in the correct order to allow mapping or each prediction in the output to a picture ID.
The total execution time for each script was between 8 and 12 days.

6 Baseline Evaluations

This section presents three points. First I present the performance of the 69 relevant humans on the evaluation data set and use their average performance to establish a human baseline. Then I present my selection process for choosing which of the 16 trained CNNs to use as my CNN baseline. Lastly I evaluate and compare this CNN baseline to the human performances and the model by Jang (2020).

The performance of each human is measured as a (FPR,TPR) pair which is calculated using equations 10 and 11. Figure 3 shows the performance of the individual humans along with a blue dot indicating the average FPR and TPR scores. This average value is not an ensemble value, but an expression of the average level of performance for the humans. It serves as a baseline for human performance for the later analysis and comparison. The average FPR is 0.196 and the average TPR is 0.765.

Each of the 16 trained models described in section 5.3 are evaluated using the ROC/AUC algorithm specified in section 4 on the training, validation, test and evaluation data sets presented in section 2. AUC scores

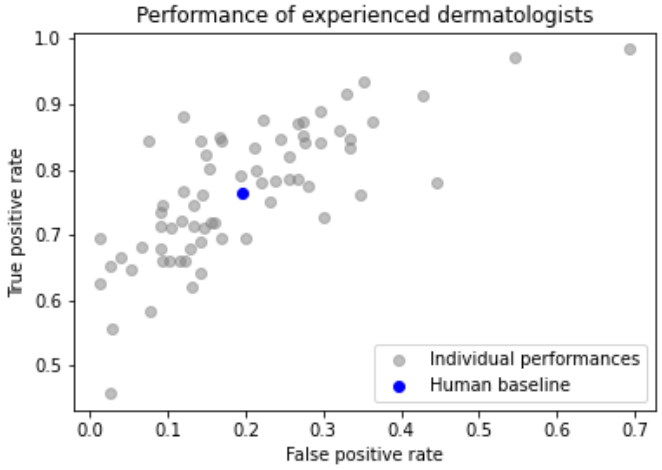

Figure 3: Performance of the dermatologists in the study. Each blue dot represents a dermatologist, and the orange dot represents the average performance.

for all models can be seen in table 9 where the highest score in each category is highlighted with boldface numbers. Visual inspection of the table reveals some interesting things. First, scores are generally highest on the training data. This is common in Machine Learning since the models have learned their features from the training data. If the training scores are much higher than the validation and test scores, this can be an indicator of overfitting.
Second, the test scores are somewhat lower than the validation scores. I see two likely explanations. One is the difference in indidence rates. The training set includes pictures from the 2019 Kaggle competition with a higher incidence rate than the testing set. If the model has learned this incidence distribution, it could cause a bias which would lead to lower test scores. Another is the difference in evaluation methods. As explained in section 4, the built-in AUC measure for the Keras package flattens the output and thus evaluates the AUC score based on all prediction values rather than only the "MEL" category. I believe the train and validation scores are falsely inflated.
Third, the evaluation data scores are lower than the testing scores. This is likely due to the qualitative differences in data sets. For one, the Kaggle data sets have been carefully processed (see section 2.1) and may have higher quality than the evaluation images. Also, the test data set can be expected to be more similar to the training images than the evaluation images are. This can lead to a bias in favor of test set performance. Secondly, the difference in incidence rate can be a factor again as the evaluation data set has a much higher incidence rate than the other data sets.
Fourth, Model 10 behaves oddly. Training performance is solid but the rest of the scores are remarkably low. Inspecting the probability outcomes show similarities to my initial model. A visual inspection on the development in AUC and loss during training shows the model trained normally for a few epochs then performance dropped severely. This can happen sometimes and I imagine it has to do with the stochastic nature of the weight initialization.

In order to select the best model, I look primarily for performance on novel data which means test and evalaution set performance. With these criteria, Model 11 is the best performer. In addition to showing the best novel data performance, it shows similar performance on the training and validation sets which would indicate no signs of overfitting. Visually inspecting the Tensorboard output in figure 4 shows a model which

ID	Labels	Depth	Dropout	train_auc	val_auc	test_auc	eval_auc
1	Binary	Deep	0	**0.995**	0.940	0.868	0.806
2	Binary	Deep	0.2	0.99	0.948	0.895	0.776
3	Binary	Deep	0.4	0.987	0.931	0.807	0.801
4	Binary	Deep	0.6	0.988	0.946	0.883	0.746
5	Binary	Shallow	0	0.971	0.912	0.751	0.705
6	Binary	Shallow	0.2	0.990	0.940	0.824	0.799
7	Binary	Shallow	0.4	0.994	0.950	0.885	0.763
8	Binary	Shallow	0.6	0.990	0.950	0.886	0.795
9	Multiclass	Deep	0	0.986	**0.983**	0.877	0.809
10	Multiclass	Deep	0.2	0.974	0.690	0.549	0.593
11	Multiclass	Deep	0.4	0.984	0.981	**0.897**	**0.822**
12	Multiclass	Deep	0.6	0.981	0.978	0.885	0.762
13	Multiclass	Shallow	0	0.981	0.927	0.766	0.762
14	Multiclass	Shallow	0.2	0.978	0.934	0.713	0.747
15	Multiclass	Shallow	0.4	0.986	0.981	0.863	0.765
16	Multiclass	Shallow	0.6	0.983	0.980	0.886	0.737

Table 9: Description and performance of the various models. Labels indicate whether the target variable is binary or multiclass. Depth refers to the capacity of the head of the network. Dropout refers to the strength of the dropout layer. All numbers are AUC scores for the respective data set. The best performer in each category has been highlighted in bold face

quickly learns the training data but spends around 17 epochs before obtaining a rather steady performance on the validation set. It then converges nicely and shows similar performances on both AUC score and loss function between training and validation data. This model seems to be the best of my trained models and is referred to as the baseline CNN model for this thesis.

On the test set, the baseline CNN achives an AUC score of 0.897 which is by no means a strong performance in the Kaggle competition field. It corresponds to a finish in the 37^{th} percentile from the bottom and is far from the winning score of 0.949.
On the evaluation data set, the baseline CNNs AUC score of 0.822 outperforms the model by Amy Jang which scores 0.773. Comparing to the human baseline performanes, the CNN baseline is inferior. At similar FPR rates, the average human has TPR = 0.765 while the CNN has TPR = 0.694. At similar TPR rates, the average human has FPR = 0.196 while the CNN has FPR = 0.320. A comparison of the two mentioned CNN models and the human average can be seen in Figure 5.

Comparing the final model with the initial training outcomes, we see a vast improvement in model performance and behavior. The final model has only 201,188 parameters which is much lower than the 11,840,195 parameters in the initial training model. Table 10 reveals the final model only has 201,188 parameters in the model head which is much lower compared to the 11,840,195 parameters in the initial model. Further, table 11 shows how the final model has a high range in the prediction values for almost all categories. This poses a sharp contrast to the narrow ranges of the initial model shown in table 8 in section 5.1. This is a clear indicator that the final model is able to distinguish features in the picture rather than just learning the training distribution.

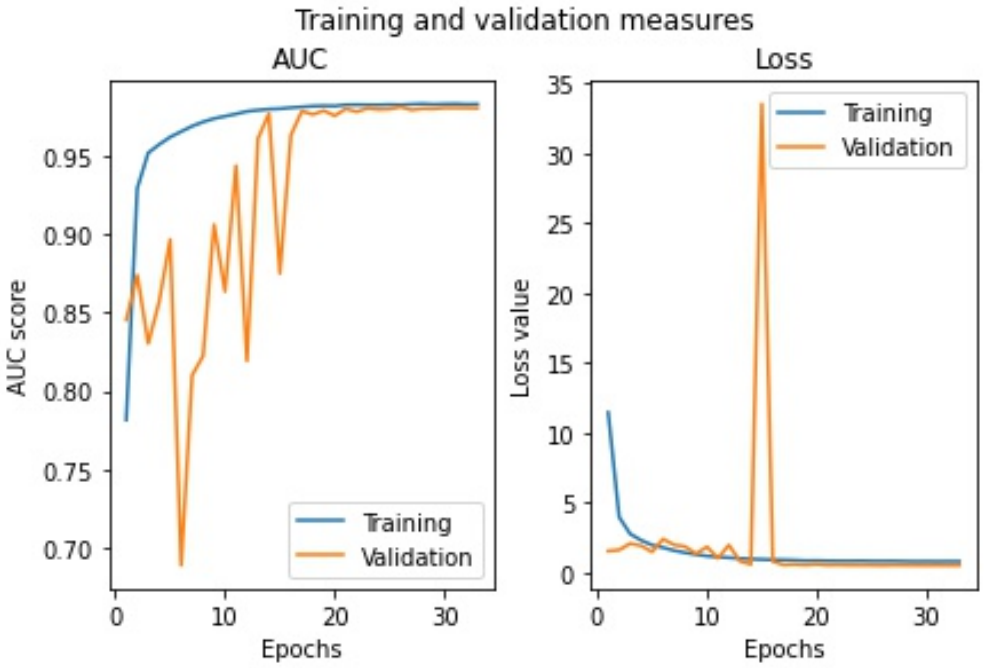

Figure 4: Tensorboard output showing training and validation AUC and loss for the best performing model.

Layer	*Output shape*	*Number of parameters*
efficientnetb3 (Functional)	(None,1536)	10783535
dropout (Dropout)	(None,1536),	0
dense_6 (Dense)	(None,128)	196763
dense_7 (Dense)	(None,32)	4128
dense_8 (Dense)	(None,9)	297

Table 10: Model summary of the top performing model.

	Category								
Label	0	1	2	3	4	5	6	7	8
Min	4.5e-11	9.3e-12	3.8e-09	2.6e-11	3.9e-07	2.0e-6	1.4e-11	7.5e-08	1.1e-11
Median	5.5e-06	2.4e-06	6.1e-05	2.3e-06	6.6e-04	3.9e-3	1.8e-06	0.99	1.8e-06
Max	0.94	0.59	0.15	0.75	0.996	1.0	0.29	1.0	0.15

Table 11: Descriptive statistics of the probability estimates from the top performing model on the test set.

To summarize, my baseline CNN seems to have converged nicely. It outperforms the imported model from Jang (2020) but falls short of the average human performance in the study. The next section will introduce and evaluate my hybrid algorithms which are build on the predictions from the human and CNN baseline.

7 Hybrid algorithms

Having established baseline performances from humans and algorithms, the attention turns to methods for ensembling these performances into hybrid approaches. The following sections introduce two approaches

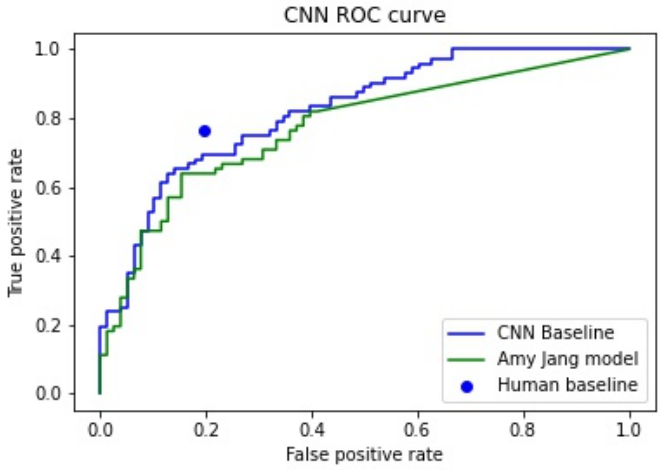

Figure 5: Figure showing the ROC curves for the CNN baseline model and Amy Jangs model. The blue dot indicates the fpr and tpr scores of the human baseline.

to hybrid ensembling. Both methods combine human and CNN predictions, but the way in which the predictions are combined differ between the methods. As a result, the evaluations of the methods look differently and the performances are different.

7.1 Augmented hybrid approach

Some pictures are harder to classify than others and it may not be the same pictures that humans and machines struggle with. S. S. Han et al. (2020) showed that some of the pictures in his data set proved difficult for the human subjects but came out with high certainty and performance scores when run through the algorithm. Similarly, Marchetti et al. (2020) showed that replacing those human answers associated with low certainty with a CNN prediction improved performance. I hypothesize that this effect goes both ways, meaning that just as some pictures are difficult for humans but easy for computers, other pictures may be easy for humans but difficult for computers. I also hypothesize that the CNNs prediction values can be used as a proxy to distinguish which images it finds "easy" and which it finds "difficult". If this is true, I can build an imputed list of human and computer predictions based on the CNNs prediction certainties. In the following sections, I introduce my algorithm and evaluate the assumption that it creates an "easy" and a "difficult" subset as measured by CNN performance. I then evaluate the algorithms performance using the baseline CNN predictions and the human predictions on the evaluation data set.

7.1.1 Augmented hybrid algorithm

Assume an array of prediction values $\boldsymbol{A} = [a_1, a_2, \ldots, a_n]$ where n is the number of pictures evaluated. Sort this array ascending so $\boldsymbol{B} = [b_1, b_2, \ldots, b_n]$ and keep track of the mapping between the index values between A and B. Introduce parameter s and define i and j as

$$i = \left\lfloor \frac{n}{s} \right\rfloor \tag{15}$$

$$j = \left\lfloor \frac{(s-1)n}{s} \right\rfloor \tag{16}$$

Note that s is constrained as $s > 2$ to ensure that $i < j$. We can now partition $\boldsymbol{B}$ into three lists

$$\boldsymbol{B_{lower}} = [b_1, b_2, \ldots, b_i] \tag{17}$$

$$\boldsymbol{B_{inner}} = [b_{i+1}, b_{i+2}, \ldots, b_j] \tag{18}$$

$$\boldsymbol{B_{upper}} = [b_{j+1}, b_{j+1}, \ldots, b_n] \tag{19}$$

Concatenate $\boldsymbol{B_{lower}}$ with $\boldsymbol{B_{upper}}$ and call this $\boldsymbol{B_{outer}}$

$$\boldsymbol{B_{outer}} = [b_1, b_2, \ldots, b_i, b_{j+1}, b_{j+2}, \ldots, b_n] \tag{20}$$

This array represents the values from $\boldsymbol{B}$ which exhibit greatest "certainty" as defined by being close to either 0 or 1. Vice versa, $\boldsymbol{B_{inner}}$ represent the values from $\boldsymbol{B}$ with the least certainty, i.e. closer to 0.5.

Algorithm 2: Augmented intelligence hybrid algorithm

input : CNN predictions, Human predictions, Ground truths, Subset index list, Number of iterations, threshold list

output: ROC Curve, AUC score

```
ROC_Curves = 3DTensor
AUC_Scores = list
foreach iteration do
    Create substituted prediction list:
    sub_predictions = list
    foreach index in CNN prediction do
        if index in Subset index list then
            Pick random human prediction for corresponding picture
            sub_predictions[index] ← humanpredictions[index]
        else
            sub_predictions[index] ← CNNpredictions[index]
    end
    Create ROC Curves & AUC Scores:
    ROC, AUC ← ROC/AUCAnalysis(input = (sub_predictions, Groundtruths, thresholdlist) # See
     algorithm 1
    UpdateROC_Curves ← ROC (stacking the dataframes on top of each other)
    UpdateAUC_scores ← AUC
end
Generate average ROC Curve:
foreach cell c_ijn in ROC_curve do
    c_ij ← (1/N) Σ_{n=1}^{N} c_ijn
end
Plot = Lineplot showing Sensitivity and Specificity measures
Average AUC ← (1/N) Σ_{n=1}^{N} AUC_Scores
Return Plot, Average AUC
```

Once the inner and outer lists have been created, the image indexes for one of them can be fed into algorithm 2 as the argument "Subset index list". This algorithm replaces the CNN predictions for the Subset index list

with randomly selected human answers for those pictures (line 4-11) into a substituted predictions list. This list is then analyzed using a normal ROC/AUC analysis as introduced in section 4. Because the algorithm picks a random human prediction for each index, stochasticity is introduced and each trial will yield different results. This can be solved by running the setup many times, as the strong law of large numbers states that the expected outcome from an infinite amount of trials will converge to the population parameter (Ross, 2002, 17).

$$\lim_{n \longrightarrow \infty} \frac{X_1 + \cdots + X_n}{n} = \mu \tag{21}$$

I run 1000 trials and append the ROC curves and AUC scores of each, after which I average the values (line 18-23 in algorithm 2). My approach to averaging the ROC curves is treating each curve as a DataFrame made up of (FPR,TPR) pairs. I stack the dataframes on top of each other to create a 3D Tensor where each cell can be represented as c_{ijn} where i represents the row number, j represents the column number and n represents the "depth" dimension corresponding to the iteration number. I then average the values along the depth dimension to get a DataFrame where

$$c_{ij} = \frac{1}{N} \Sigma_{n=1}^{N} c_{ijn} \tag{22}$$

I use this DataFrame to draw the resulting ROC curve. Getting the average AUC is done by taking the arithmetic mean of the AUC scores

$$AUC\_average \leftarrow \frac{1}{N} \Sigma_{n=1}^{N} AUC\_Scores \tag{23}$$

7.1.2 Testing differences between humans and CNN

The augmented intelligence hybrid algorithm builds on the assumption that humans and CNNs perform well on different pictures and that their performances are not highly correlated. If their performances have high correlation, substituting the guess of one with with the other should make little or no impact. To test this assumption, I split the CNN predictions into an inner and an outer list in accordance with equations 18 and 20. I use $s = 4$ as this makes the lists equally big. Both lists are evaluated by both the baseline CNN model and the human answer. The results, which can be seen in figure 6, show signs that some images are easier to identify than others. It can be seen that the CNN performance is worst for the inner list which has AUC = 0.664 and best on the outer list which has AUC = 0.917. Similarly, the humans seem to perform worst on the inner list and best on the outer list. Their TPR score changes between the lists while the FPR score is fairly unchanged. This support the hypothesis that some pictures are harder to classify than others. It also indicates that we can use the CNN predictions as a proxy to isolate these pictures.
For the inner list, the CNN performance is far inferior to the human performance, but on the outer list the CNN performs slightly better than the humans. This supports the hypothesis that humans and CNNs have difficulties with different pictures and thus their performance is not highly correlated.

7.1.3 Augmented hybrid results

Having shown that some pictures are harder to classify than others and that human and CNN performance is not highly correlated, the next step is creating a merged list of human and CNN answers. Since CNNs are better on the outer list and humans are better on the inner list, I hypothesize that a prediction list where the inner elements consists of human answers and the outer list consists of CNN predictions will perform

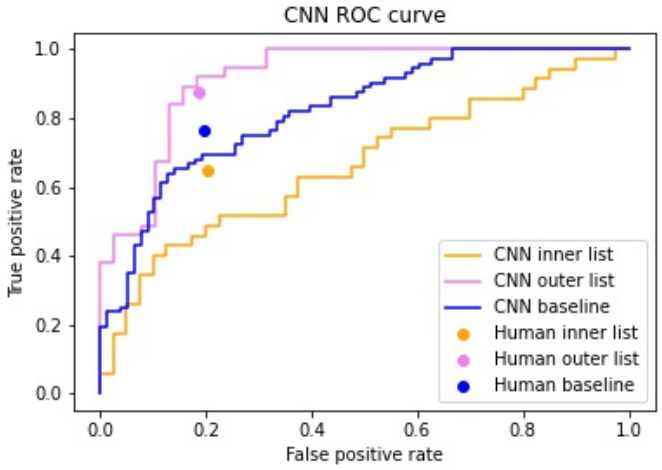

Figure 6: Human and CNN performance for the two sublists inner and outer. The baseline human and CNN performances are included as reference.

better than the baseline performances. In order to test this, I run algorithm 2 twice - once with the inner list as the "Subset index list" and once with the outer list as the "Subset index list". In both cases I run the algorithm 1000 times with the expectancy that it will converge towards the expected ROC curve and AUC values.

Figure 7 shows the result from creating and analyzing the inner and outer substitution lists. The outer substitution list generally follows the CNN baseline model except at the ends. It has an AUC score of 0.798 which is slightly below the baseline CNN score of 0.822.
The inner substitution line lies below the CNN baseline for $FPR[0, 0.17]$, then above the baseline for $FPR[0.17, 0.4]$ and then dropping below the baseline for $FPR[0.4, 1]$. It has an AUC score of 0.772 which is slightly below the CNN baseline. However, the inner substitution line has higher TPR (0.782) than the human baseline given similar FPR and likewise lower FPR (0.182) than human baseline given similar TPR. This can support a claim that the inner substitution model is superior to the human baseline and will be discussed in more depth in section 8.

7.2 Hybrid majority vote approach

Sampling multiple observations from a group and taking the majority vote outperforms the group average and can sometimes even outperform the strongest group member (R. Kurvers et al., 2019, 2015; R. H. J. M. Kurvers et al., 2016). However, the cost of achieving this extra certainty is high which I explain in the next sections. My hypothesis is that if humans and CNNs perform approximately similarly, we can include CNN predictions into majority votes and achieve similar results as we would with human predictions. With the assumption that taking a picture and running it through a CNN is cheaper than asking a human being, my algorithm is a cheaper solution than building human-only majority votes. In the following sections I first test the premise that a human majority vote outperforms the human average on the evaluation data. I then explain how to algorithmically include the CNN predictions in a hybrid majority vote. Lastly I show the results from combining human and CNN predictions and compare the performance and cost associated with that of a human only approach.

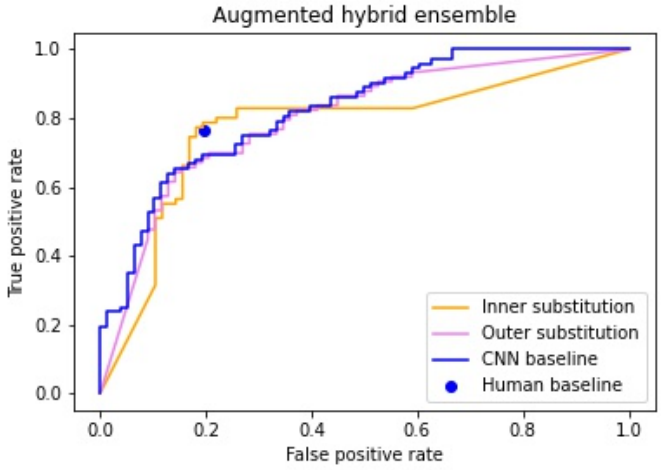

Figure 7: Graph showing the results of averaging 1000 simulations of the inner and outer hybrid lists. The blue dot and line show the baseline human and CNN performance.

7.2.1 Human ensembling

Performing majority vote ensembling in a binary setting can be done fairly simply. Let

$$g_i, \quad i \in (1, 2, \ldots, N) \tag{24}$$

represent the i'th guess for an group of size N where N is an odd number. Let $\boldsymbol{G}$ represent the ensemble answer. Since the setting is binary, $g0, 1$. We can now calculate the ensemble answer as

$$\boldsymbol{G} \begin{cases} 0 & \frac{1}{N}\Sigma_{i=1}^{N} g_i < 0.5 \\ 1 & \frac{1}{N}\Sigma_{i=1}^{N} g_i \geq 0.5 \end{cases} \tag{25}$$

I use this method to calculate the full ensemble performance shown in Figure 8. However, the method does not give information regarding the cost function because it assumes asking all N participants which may not be necessary. So for the $N = 3$ ensemble in Figure 8, I expand on the approach using the assumption described below. This allows me to record how many votes are necessary for each guess.

In a binary setting, a majority is achieved when one option has at least $\lceil \frac{N}{2} \rceil$ votes. For a three man ensemble ($N = 3$), if $g_1 = g_2 \longrightarrow g_1 = g_2 = \boldsymbol{G}$ regardless of g_3 and if $g_1 \neq g_2 \longrightarrow g_3 = \boldsymbol{G}$. This means the expected number of guesses until majority has been achieved will lie in the range $[2, 3]$. By counting how many times $g_1 \neq g_2$, I know how many times the third opinions is necessary and thus how many votes are needed on average for each guess.

I record the performance of the human majority vote ensembles for $N = 69$ and for $N = 3$. Randomly selecting three opinions at a time generates a stochastic variable, so I average the results from 1000 simulations to approximate the expected outcome (Ross, 2002). The results of performing human majority votes can be seen in Figure 8. We see a perfect score for the full ensemble consisting of all 69 humans. For the three-man ensemble, I get (FPR,TPR) scores of (0.150,0.787) which outperforms the human baseline. This method

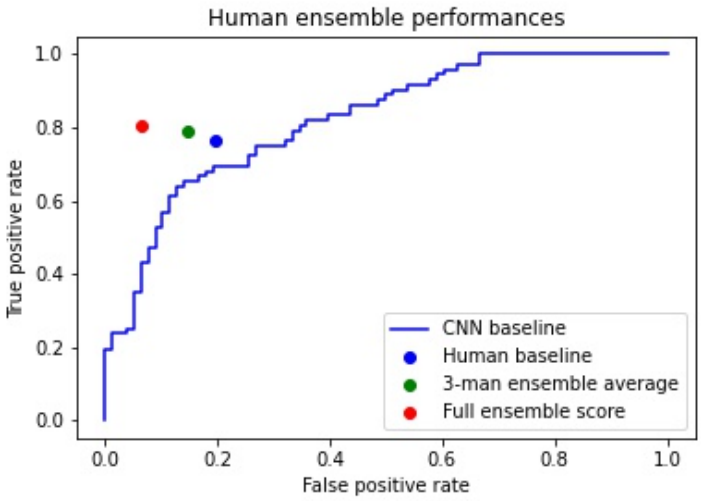

Figure 8: Performance of the human majority vote ensembles compared to the baseline. The green dot represents the average scores from 1000 trials with three man ensembles. The red dot represents the full ensemble score.

requires on average 35.5 third opinions for the 150 images meaning the average number of humans necessary for each picture is $2 + \frac{35.5}{150} = 2.24$.

The results support the theory that human ensembling can improve performance on this evaluation data set. The next section introduces a method for including a CNN prediction into the majority vote ensemble method with $N = 3$.

7.2.2 Hybrid majority vote algorithm

To create a $N = 3$ hybrid majority vote, let g_1 and g_2 represent guesses from two randomly selected humans on a picture, let p_{alg} represent the probability prediction from a computer model and let $\boldsymbol{G}$ represent the ensemble guess. Further let $t \in \boldsymbol{T}$ represent a threshold. Now create g_{alg} as

$$g_{alg} \begin{cases} 0 & p_{alg} < t \\ 1 & p_{alg} \geq t \end{cases} \tag{26}$$

If $g_1 = g_{alg} \longrightarrow g_1 = g_{alg} = \boldsymbol{G}$ regardless of g_2 and if $g_1 \neq g_{alg} \longrightarrow g_2 = \boldsymbol{G}$. Since one of the opinions in each ensemble is from a human, the expected number of human opinions for the hybrid majority vote lies in the range $[1, 2]$.
Because p_{alg} is a decimal number in range $[0, 1]$, the value for each picture g_{alg} will depend on the threshold t. Low values of t will resuls in many pictures being classified as positive and high values of t will result in few pictures being classified as positive. This means the expected outcome of a trial will depend on the threshold. To make a fair comparison with the baseline CNN performance, I use same thresholds for this algorithm as I use to establish the CNN baseline in section 6.

Due to the random selection of human answers, $\boldsymbol{G}$ is a stochastic random variable and the performance will vary between trials. I run and average 1000 trials for each threshold so the outcomes should approximate the expected value for each threshold (Ross, 2002). I also note and average the number of third opinions

by counting how often $g_1 \neq g_{alg}$. An algorithmic explanation of the hybrid majority procedure is shown in Algorithm 3.

Algorithm 3: Majority vote hybrid approach

input : CNN predictions, Human predictions, Ground truths, Image list, Number of iterations, threshold list

output: ROC Curve, AUC score

```
1  Scoring = dict
2  Third_Opinions = list
3  foreach threshold in threshold list do
4      t ← threshold
5      FPR_values = list
6      TPR_values = list
7      Third_Opinion_values = list
8      foreach iteration do
9          New_predictions = list
10         Third_Opinions_Count ← 0
11         foreach index in index list do
12             p_alg ← CNN_prediction[index]
13             if p_alg > t then
14                 g_alg = 1
15             else
16                 g_alg = 0
17             g_1, g_2 ← randomly select two answers from Human predictions without replacement
18             if g_alg = g_1 then
19                 pred ← g_1
20             else
21                 pred ← g_2
22                 Third_Opinions_Count ← +1
23             Update New_predictions ← pred
24         end
25         FPR,TPR ← Calculate from New_predictions and Ground Truths using line 9-23 in algorithm 1
26         Update FPR_values ← FPR
27         Update TPR_values ← TPR
28         Update Third_Opinions_values ← Third_Opinions_Count
29     end
30     Average_FPR ← (1/N) ΣFPR_values
31     Average_TPR ← (1/N) ΣTPR_values
32     Average_Third_Opinions ← (1/n) ΣThird_Opinions_Values
33     Update Scoring[t] ← (Average_FPR, Average_TPR)
34     Update Third_Opinions ← Average_Third_Opinions
35 end
36 Return Scoring, Third_Opinions
```

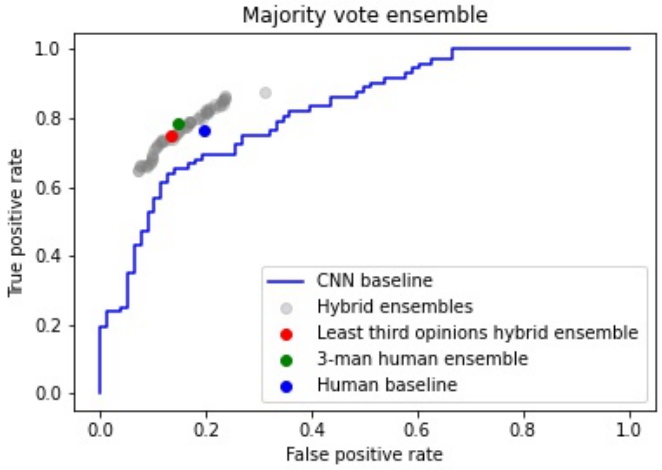

Figure 9: Hybrid majority ensemble results. The blue line and dot represents the CNN and human baseline respectively. The green dot represents the three man human ensemble. The grey dots represent the hybrid majority vote performances for each threshold. The red dot represents the hybrid majority vote which uses a third opinion the least amount of times.

7.2.3 Hybrid majority vote performance

The results from the hybrid majority vote can be seen in Figure 9 compared to the baseline performances and the three human ensemble presented in the previous section. The grey dots represent the (FPR,TPR) scores for each threshold in $\boldsymbol{T}$. The red dot highlights the threshold which is associated with the lowest cost as measured by third opinions necessary. It represents (FPR,TPR) values of (0.135,0.747). Compared to the three human ensemble, this is slightly lower FPR and slightly lower TPR. Compared to the human baseline, the TPR is similar and the FPR is lower which would suggest better performance. The red dot represents a threshold which uses on average 38.5 third opinions. This means the algorithm given this specific threshold requires on average $1 + \frac{38.5}{150} = 1.26$ human opinions per image to achieve majority.
Visual inspection of the grey dots in figure 9 reveal a somewhat straight line which lies above the human and CNN baseline values. It appears to perform closely to that of the three-human ensemble. At a similar TPR-value to the three-man ensemble, the hybrid ensemble achieves an FPR value of 0.169 which is slightly higher. This is achieved using 1.40 opinions on average per picture. At a similar FPR value to the three-man ensemble, the hybrid ensemble achieves a TPR value of 0.765 which is slightly lower. This is achieved using 1.35 opinions on average per picture. The highest achievable TPR with this method is 0.876 which comes at an FPR of 0.312 and with a cost of 1.53. The lowest possible FPR with the method is 0.072 which comes with a TPR of 0.650 and at a cost of 1.47. An overview of all mentioned human and hybrid majority vote performances and associated costs can be found in table 12.

8 Discussion

With the high prevalence and mortality rate from skin cancer (Rogers et al., 2015), quick and accurate diagnosis is paramount. As discussed in section 1, much research has attempted to solve this. Aggregation approaches often improve predictive performance. In this study, I have found that taking a majority vote of 69 medical professionals gives perfect performance; however, asking so many professionals comes with a high cost and does not seems to be a viable solution. Other research has shown that using computers for

Ensemble type	Specific type	FPR	TPR	Cost
Human only	Human baseline	0.196	0.765	1
	Three-human majority	0.150	0.787	2.24
	All human majority	0.064	0.806	69
Human-computer	Min cost hybrid ensemble	0.140	0.750	1.26
	Minimum FPR	0.072	0.650	1.47
	Maximum TPR	0.312	0.876	1.53
	Matching three-human FPR	0.150	0.765	1.35
	Matching three-human TPR	0.169	0.789	1.40

Table 12: Overview of the FPR, TPR, and cost values for the baseline and majority vote models. The human baseline represents the average human performance as described in section 6. The Three-human ensemble has no computers involved and is presented

these tasks can achieve better performance than humans. In this study, I the computer model is inferior to human performance. Hybrid approaches have proven to be a way of improving performance, but research into the costs and ethical considerations of doing this is lacking. In this study, I present and explore two approaches to hybrid ensembling which can improve performance at a low cost while keeping the human in the loop. In the following sections, I discuss the methods results, costs, limitations, and future prospects.

8.1 The augmented hybrid approach

The augmented hybrid approach which substitutes the inner values could be an interesting topic of further research. Despite achieving a lower AUC score of 0.769 compared to the baseline score of 0.822, I argue the resulting ROC curve is interesting and may in fact be useful in a real world application. Setting a threshold for the augmented ensemble which gives an FPR similar to the baseline human performance gives a slightly higher TPR than the human baseline performance. Similarly, setting a threshold which gives the same TPR as the human baseline gives a slightly lower FPR compared to the human baseline level. This evaluation method is a common approach to compare human v. algorithmic performance in the literature (T. Brinker et al., 2019; S. S. Han et al., 2020) and would suggest the approach has slightly superior performance to the human baseline.

I suspect the reason for the lower AUC score of the inner substitution approach lies in the binary nature of the human answers. The algorithm imputes the elements on the inner list with either "0" or "1". Since the elements on the outer list consist of numbers between 0 and 1, this approach effectively makes the predictions on the inner list more "certain" than the elements on the outer list. And since we know the elements on the inner list are more difficult to classify, this will diminish performance when $t = 0 \wedge t = 1$ compared to a baseline approach. This reasoning would explain why the graph representing the inner list on Figure 7 has poor performance towards the ends.

For the purpose of strict comparison, it may not seem fair to compare individual humans to humans aided by computers. However, my desire with this algorithm is to provide a minimum-cost solution which improves diagnostic performance. The world is not ready to take the human out of the decision loop (Haggenmüller et al., 2021; S. S. Han et al., 2020; Navarrete-Dechent et al., 2021), so I assume the minimum number of doctors involved in a decision must be one. My approach represents a way to enhance the performance of a single doctor with minimal associated costs and should not be seen as an attempt to trivialize the performance of

medical professionals.

The generalizeability of the method is difficult to determine from only this one study. As mentioned in section 8, the baseline performance of the CNN is inferior to the baseline human performance in this study, and this is unusual compared to the literature. We also don't know if the enhanced performance is merely a product of the specific data set or if it represents an underlying superior performance. However, I present now an intuition to what I think drives the performance enhancement.
The model isolates two sublists from the image list. Depending on the relative performance of the humans and the CNN on those lists, there are three potential outcomes. In the first outcome, the CNN outperforms the humans on both lists. In this case, the ensemble should perform worse than the CNN baseline but better than the human baseline. In the second outcome, the humans outperform the CNN on both lists. In this case, the ensemble should perform better than the CNN baseline but worse than the humans. In the third outcome, one group is superior on one list but inferior on the other. In this case, which representative of this study, the ensemble is able to take the superior human performance from one list and combine with the superior CNN performance on another list. It basically utilizes the strengths of both groups and combines it into an even stronger ensemble.

Based on the above discussion, I believe this model may generalize under the criteria that humans are superior on one list and the CNN is superior on another list. This may not always be the case, especially when using SOTA computer vision models as those may beat the humans on both lists. In such case it may be worth exploring the parameter s introduced in section 2 which determines the relative sizes of the lists. I have gotten results using $s = 4$ in this study and have not explored its implications in depth. A quick analysis reveals that exploring different values of s gives different inner and outer lists which results in different baseline performances along with different performance from this algorithm. I provide a visual representation of this in Figure 12 in section 11.4. So if working with a SOTA CNN which is superior on both lists, I will recommend exploring the value of s and see if this creates a situation where the humans are better on one list. In this case, my hybrid algorithm may be able to outperform even very strong CNNs.
I realize that using s to create the inner and outer lists only makes sense using multiple observations. If the network is presented with a single, novel observation the value of s and subsequently equations 15 and 16 are useless. For this reason, s can only be used as a parameter to discover different ratios between the inner and outer lists to find a suitable configuration. To make the algorithm production ready, I suggest finding the corresponding thresholds which divide the inner and outer lists similarly to i and j. A new image could then be evaluated and assigned quickly to the correct list by comparing the prediction value to the pre-determined thresholds.

I suggest the following procedure could be implemented in medical offices:

- The medical professional makes his/her evaluation as usual.
- The medical professional opens an app, inputs his/her evaluation and takes a picture of the lesion.
- The app sends the image to a cloud service which runs the image through a CNN. The prediction is then compared to a predetermined lower and upper threshold to determine if it falls in the inner or outer list.
- If the image falls on the inner list, the app returns the doctors own opinion to the doctor.
- If the image falls on the outer list, the app returns the CNNs prediction to the doctor and the doctor is allowed to make the final decision.

I realize the last step in this approach is slightly different from the approach described in this thesis where the doctors opinion was overruled by the CNN prediction. My proposed alteration originates from ethical considerations. It may decrease performance slightly, but I believe this will pose a smart trade-off.

To summarize, the augmented hybrid approach combines the strengths of the humans and CNN to create an improved performance at a minimum cost since it does not involve any extra humans. This study is limited by a relatively poor CNN performance and the results may not generalize well. The only way to know is to test the method using SOTA CNN models on other data sets which also have the human predictions recorded. When doing so, I recommend treating s as a parameter and investigate its implications.

8.2 The majority vote hybrid approach

The majority vote hybrid approach may be a low cost solution which can save lives. My results indicate that building a three piece ensemble consisting of two humans and one CNN model performs better than the baseline of humans or CNN separately. This ensemble performs nearly as well as a three human ensemble despite the inferior performance of the algorithm relative to the humans, and it comes with a relatively low cost. I hypothesize that if the CNN had superior performance to the human baseline, the results of the ensemble would have beaten the three-human ensemble. Further research using stronger CNNs are needed to validate if this hypothesis is true.

A potential critic on the generalizeability of the method is that if the CNN is far superior to humans, the method may be useless as R. Kurvers et al. (2019) showed the individual performances need to be similar. However, that study looks at the ability of the ensemble to beat the best individual performance. It showed that the ensemble performance may not beat the best individual performance. So if we assume a situation where the CNN is far superior to humans, the result of the majority vote approach will likely not beat the performance of the CNN baseline. However, this approach should still always beat baseline human baseline performance (see Appendix section 11.2 for my reasoning behind this argument). So until we are ready to take humans out of the loop, using this algorithm with a far superior CNN is a cheap solution to get better results.

I have shown in Figure 12 that different thresholds produce different performances at different costs when using the hybrid majority vote algorithm. Getting a high TPR should definitely be a priority, but maximizing TPR comes with higher FPR and higher human resource costs. In this project, the expected cost is between $1.26 - 1.53$. As such, there may be some economic principle to find the optimal threshold, but finding this goes far beyond the scope of current thesis.

I suggest the following procedure could be implemented in medical offices:

- The medical professional makes his/her evaluation as usual.
- The medical professional opens an app, inputs his/her evaluation into the app and takes a picture of the lesion.
- The app sends the image to a cloud service which runs the image through a CNN and evaluates it using a pre-determined threshold. The answer is sent back to the app.
- The app now compares the input from the medical professional and the CNN and determines if the two agree. If they do not, a message is sent to another medical professional to come and make an independent evaluation of the results.

In the last step of the procedure, the other medical professional is not supposed to know the answer from the first human not the machine as this may cause some bias. Instead, the other professional will evaluate as usual and this answer will be treated as the verdict. Following the procedure will, based on the results from this study, improve medical diagnosis with relative low cost. The solution also does not take the human out of the loop as the final decision will always be one which a human has voted for.

In summary, the majority hybrid algorithm is a cheap implementation which seems to improve diagnostic performance in medical offices and could in that sense save lives at a low cost. I have shown that it improves performance even with inferior CNN performance and provided a theoretical framework for why it will likely also work if the CNN performance is superior. I have sketched a hands-on approach to how the method could be implemented. However, I think the method needs more testing with larger data sets and SOTA CNNs to verify my assumptions regarding performance in this setting.

8.3 Comparison between approaches

From a performance standpoint, both hybrid approaches show promising results as means to implementing low-cost improvements to the dermatologist work. The augmented hybrid approach slightly outperforms the human baseline in this study while incurring minimal cost as it only requires the medical professional to take a picture with an app. The majority vote hybrid approach clearly outperforms the human baseline and performs closely to a three-human ensemble while costing almost one less human guess on average. Compared to the augmented hybrid approach, it has a slightly higher cost. So for this study, the majority vote ensemble seems to be the superior but also most expensive performer. However, this study is not quite representative as the CNN is weak relative to the humans. I suspect that in situations where the CNN outperforms the humans, we might see that the augmented hybrid approach performs stronger than the majority vote hybrid approach. More investigation is necessary and I would recommend both approaches are tested further with SOTA computer models on other data sets to see if these tendencies remain intact.

From an ethical standpoint, the majority vote may be the most viable approach. By using this approach, a patient is ensured that the diagnosis prescribed is recommended by at least one doctor. In the augmented approach, the ethical consideration depends on whether the CNN is allowed to overrule the doctors opinions or the doctor gets the final answer. Letting the CNN overrule, unless the CNN has $TPR = 1$ in these situations, there will be False Negatives and this will likely cause poor media attention. And since we can never be certain that $TPR = 1$ on novel data, this branch of research can probably not be implemented until the general public is ready to trust AI with their lives. So from an ethical standpoint, the majority vote approach is probably the approach which could be implemented sooner.

In summary, the hybrid majority vote algorithm is superior in terms of both performance and ethics, but it comes at a higher cost in terms of human resources. The superior performance may not be generalizable, but the ethical advantage and higher cost must generalize as they are related to the method itself and not the data used in this study.

8.4 Final remarks

Before concluding this thesis I want to point out two final remarks. The first one is a general observation which is relevant when discussing whether a CNN model is superior to humans in a given study. The second one is a mistake which I found in my script.

The literature review for this thesis has revealed several instances where CNNs outperform humans on skin lesion classification tasks. However, none of these studies seem to question whether their human sample

performance is representative. Reviewing some of the literature used for this study reveals high discrepancy between the human performance levels. These differences could be due to differences in the expertise of the humans or difficulties in the pictures used. Regardless, I think it is worth paying attention to the representativeness of the human samples when looking at studies where the CNN has superior performance. Table 13 shows an overview of select human performances from the literature used in this thesis. Of the seven studies listed, this thesis has the second lowest FPR and the median TPR. The two human sample performances in the bottom of the table perform worse than the CNN baseline in this study.

Study	FPR	TPR
H. A. Haenssle et al. (2020)	0.21	0.89
H. Haenssle et al. (2018)	0.29	0.86
S. S. Han et al. (2020)	0.07	0.77
This thesis	0.20	0.77
Marchetti et al. (2020)	0.28	0.76
T. Brinker et al. (2019)	0.4	0.74
T. J. Brinker et al. (2019)	0.38	0.67

Table 13: A sample of human sample performances compared to the performance in this thesis. The table is sorted descending using the TPR column. The table shows a great range in the average human performance between the studies.

I found a mistake is my code which may affect the performance of the CNN models and cause them to perform sub-optimally. When training CNNs, it is customary to pre-process the images and downscale the color channel by a factor of 255 (actually 256 but Python is a zero-indexed programming language) so it falls in range $[0, 1]$. I do this equally for all images, meaning both the training, validation, testing and evaluation images. After training and evaluating the models, I realized the documentation for EfficientNets says: "For EfficientNet, input preprocessing is included as part of the model (as a Rescaling layer), and thus tf.keras.applications.efficientnet.preprocess_input is actually a pass-through function. EfficientNet models expect their inputs to be float tensors of pixels with values in the [0-255] range." (Team, n.d.). This means the pictures in my study ultimately have been rescaled twice into the $[0, \frac{1}{255}]$ range. Re-training the models was not an option so the final results include this mistake. I suspect the effect of it will only be minor, but future researches should beware of making my mistake when working with EfficientNets. The mistake has been left in the code but commented out.

9 Conclusion

The baseline results show an inferior performance of the CNN compared to human baseline which is unusual for this field. However, the two presented hybrid approaches outperform both human and CNN baseline performances with low cost. The augmented hybrid approach outperforms the human and CNN baselines while keeping the cost at a minimum. The majority vote hybrid approach outperforms human and CNN baseline strongly with a slight increase in human cost. The majority method almost matches the performance of a three human ensemble, but with a much lower associated cost.
The study is limited by a CNN which is far from state of the art and performs worse than human baseline which is unusual for the field of study. I theorize that a stronger CNN would be an advantage to both algorithms and create better performances. The algorithms may not outperform the baseline CNN performance when using a SOTA, but they will be an ethically defendable approach which outperforms human performance at a low cost.
With the results and considerations presented in this study, I believe the algorithms provide a starting point for improving diagnostic performance in medical offices. The augmented algorithm may run into ethical problems as its pure form allows a machine to make potential life-important decisions; however, this can be adjusted to give a human the final vote. The majority vote algorithm may prove implausible for financial or practical considerations as it requires the involvement of more than one human on average.
I have proposed two models which may be applicable in real wold situations and save lives; however, I reckon they need to be tested in depth with SOTA models and novel data. In addition research into parameter tuning would help and considerations regarding cost should be made before employing either method in a real world setting.

10 Literature

Alhichri, H., Alsuwayed, A., Bazi, Y., Ammour, N., & Alajlan, N. (2021, 01). Classification of remote sensing images using efficientnet-b3 cnn model with attention. *IEEE Access*, *PP*, 1-1. doi: 10.1109/ACCESS.2021.3051085

Bo. (2020, Aug). *Melanoma-winning-models.* Retrieved from `https://www.kaggle.com/datasets/boliu0/melanoma-winning-models`

Boland, P. J. (1989). Majority systems and the condorcet jury theorem. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *38*(3), 181–189.

Brinker, T., Hekler, A., Enk, A., Klode, J., Hauschild, A., Berking, C., ... Collaborators (2019, 04). Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer*, *113*, 47-54. doi: 10.1016/j.ejca.2019.04.001

Brinker, T. J., Hekler, A., Enk, A. H., Berking, C., Haferkamp, S., Hauschild, A., ... Utikal, J. S. (2019). Deep neural networks are superior to dermatologists in melanoma image classification. *European journal of cancer (1990)*, *119*, 11-17.

Cdeotte, C. (2020, Jul). *Triple stratified kfold with tfrecords.* Chris Deotte. Retrieved from `https://www.kaggle.com/code/cdeotte/triple-stratified-kfold-with-tfrecords/notebook`

Chollet, F. (2017). *Deep learning with python.* Manning.

Codella, N., Rotemberg, V., Tschandl, P., Celebi, M. E., Dusza, S., Gutman, D., ... others (2019). Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*.

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., ... others (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 ieee 15th international symposium on biomedical imaging (isbi 2018)* (pp. 168–172).

Combalia, M., Codella, N. C. F., Rotemberg, V., Helba, B., Vilaplana, V., Reiter, O., ... Malvehy, J. (2019). *Bcn20000: Dermoscopic lesions in the wild.* arXiv. Retrieved from `https://arxiv.org/abs/1908.02288` doi: 10.48550/ARXIV.1908.02288

Esteva, A., Kuprel, B., Novoa, R., Ko, J., Swetter, S., Blau, H., & Thrun, S. (2017, 01). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*. doi: 10.1038/nature21056

Galton, F. (1907). Vox populi (the wisdom of crowds). *Nature*, *75*(7), 450–451.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT press.

Gutman, D., Codella, N. C., Celebi, E., Helba, B., Marchetti, M., Mishra, N., & Halpern, A. (2016). Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1605.01397*.

Ha, Q., Liu, B., & Liu, F. (2020). *Identifying melanoma images using efficientnet ensemble: Winning solution to the siim-isic melanoma classification challenge.*

Haenssle, H., Fink, C., Schneiderbauer, R., Toberer, F., Buhl, T., Blum, A., ... Uhlmann, L. (2018, 05). Man against machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Annals of oncology : official journal of the European Society for Medical Oncology*, *29*. doi: 10.1093/annonc/mdy166

Haenssle, H. A., Fink, C., Toberer, F., Winkler, J., Stolz, W., Deinlein, T., ... Groups, L. I. (2020). Man against machine reloaded: performance of a market-approved convolutional neural network in classifying a broad spectrum of skin lesions in comparison with 96 dermatologists working under less

artificial conditions. *Annals of oncology*, *31*(1), 137-143.

Haggenmüller, S., Maron, R. C., Hekler, A., Utikal, J. S., Barata, C., Barnhill, R. L., ... Brinker, T. J. (2021). Skin cancer classification via convolutional neural networks: systematic review of studies involving human experts. *European Journal of Cancer*, *156*, 202-216. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0959804921004445` doi: https://doi.org/10.1016/j.ejca.2021.06.049

Han, S., Kim, M., Lim, W., Park, G., Park, I., & Chang, S. (2018, 02). Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *Journal of Investigative Dermatology*, *138*. doi: 10.1016/j.jid.2018.01.028

Han, S. S., Park, I., Eun Chang, S., Lim, W., Kim, M. S., Park, G. H., ... Na, J.-I. (2020). Augmented intelligence dermatology: Deep neural networks empower medical professionals in diagnosing skin cancer and predicting treatment options for 134 skin disorders. *Journal of Investigative Dermatology*, *140*(9), 1753-1761. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0022202X20301366` doi: https://doi.org/10.1016/j.jid.2020.01.019

Haqishen. (n.d.). *Haqishen/siim-isic-melanoma-classification-1st-place-solution.* Retrieved from `https://github.com/haqishen/SIIM-ISIC-Melanoma-Classification-1st-Place-Solution`

Hekler, A., Utikal, J., Enk, A., Hauschild, A., Weichenthal, M., Maron, R., ... Brinker, T. (2019, 09). Superior skin cancer classification by the combination of human and artificial intelligence. *European Journal of Cancer*, *120*, 114-121. doi: 10.1016/j.ejca.2019.07.019

Hekler, A., Utikal, J. S., Enk, A. H., Berking, C., Klode, J., Schadendorf, D., ... Brinker, T. J. (2019). Pathologist-level classification of histopathological melanoma images with deep neural networks. *European journal of cancer (1990)*, *115*, 79-83.

Ho, Y.-C., & Pepyne, D. (2001). Simple explanation of the no free lunch theorem of optimization. In *Proceedings of the 40th ieee conference on decision and control (cat. no.01ch37228)* (Vol. 5, p. 4409-4414 vol.5). doi: 10.1109/CDC.2001.980896

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in r.* Springer. Retrieved from `https://faculty.marshall.usc.edu/gareth-james/ISL/`

Jang, A. (2020, Jul). *Tensorflow + transfer learning: Melanoma.* Amy Jang. Retrieved from `https://www.kaggle.com/code/amyjang/tensorflow-transfer-learning-melanoma/notebook`

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kurvers, R., Herzog, S., Hertwig, R., Krause, J., Moussaïd, M., Argenziano, G., ... Wolf, M. (2019, 11). How to detect high-performing individuals and groups: Decision similarity predicts accuracy. *Science Advances*, *5*, eaaw9011. doi: 10.1126/sciadv.aaw9011

Kurvers, R., Herzog, S., Hertwig, R., Krause, J., & Wolf, M. (2021, 06). Pooling decisions decreases variation in response bias and accuracy. *iScience*, *24*, 102740. doi: 10.1016/j.isci.2021.102740

Kurvers, R., Krause, J., Argenziano, G., Zalaudek, I., & Wolf, M. (2015, 10). Detection accuracy of collective intelligence assessments for skin cancer diagnosis. *JAMA dermatology*, *151*, 1-8. doi: 10.1001/jamadermatol.2015.3149

Kurvers, R. H. J. M., Herzog, S. M., Hertwig, R., Krause, J., Carney, P. A., Bogart, A., ... Wolf, M. (2016). Boosting medical diagnostics by pooling independent judgments. *Proceedings of the National Academy of Sciences*, *113*(31), 8777–8782. Retrieved from `https://www.pnas.org/content/113/31/8777` doi: 10.1073/pnas.1601827113

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-

based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Li, W., Zhuang, J., Wang, R., Zhang, J., & Zheng, W.-S. (2020). Fusing metadata and dermoscopy images for skin disease diagnosis. In (p. 1996-2000). IEEE.

Mahbod, A., Schaefer, G., Ellinger, I., Ecker, R., Pitiot, A., & Wang, C. (2018, 11). Fusing fine-tuned deep features for skin lesion classification. *Computerized Medical Imaging and Graphics*, *71*. doi: 10.1016/j.compmedimag.2018.10.007

Marchetti, M. A., Liopyris, K., Dusza, S. W., Codella, N. C., Gutman, D. A., Helba, B., ... Malvehy, J. (2020). Computer algorithms show potential for improving dermatologists' accuracy to diagnose cutaneous melanoma: Results of the international skin imaging collaboration 2017. *Journal of the American Academy of Dermatology*, *82*(3), 622-627. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0190962219323734` doi: https://doi.org/10.1016/j.jaad.2019.07.016

Mendonça, T., Ferreira, P. M., Marques, J. S., Marcal, A. R. S., & Rozeira, J. (2013). Ph¡sup¿2¡/sup¿ - a dermoscopic image database for research and benchmarking. In *2013 35th annual international conference of the ieee engineering in medicine and biology society (embc)* (p. 5437-5440). doi: 10.1109/EMBC.2013.6610779

Navarrete-Dechent, C., Liopyris, K., & Marchetti, M. A. (2021). Multiclass artificial intelligence in dermatology: Progress but still room for improvement. *Journal of Investigative Dermatology*, *141*(5), 1325-1328. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0022202X20321424` doi: https://doi.org/10.1016/j.jid.2020.06.040

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, *29*(9), 2352-2449.

Rogers, H. W., Weinstock, M. A., Feldman, S. R., & Coldiron, B. M. (2015, 10). Incidence Estimate of Nonmelanoma Skin Cancer (Keratinocyte Carcinomas) in the US Population, 2012. *JAMA Dermatology*, *151*(10), 1081-1086. Retrieved from `https://doi.org/10.1001/jamadermatol.2015.1187` doi: 10.1001/jamadermatol.2015.1187

Ross, S. M. (2002). *Simulation* (3rd ed.). USA: Academic Press, Inc.

Rotemberg, V., Kurtansky, N., Betz-Stablein, B., Caffery, L., Chousakos, E., Codella, N., ... Soyer, H. P. (2021). A patient-centric dataset of images and metadata for identifying melanomas using clinical context. *Scientific data*, *8*(1), 34-34.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, *6*(12), 310–316.

Siim-isic melanoma classification. (n.d.). Retrieved from `https://www.kaggle.com/c/siim-isic-melanoma-classification/overview`

Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, *abs/1905.11946*. Retrieved from `http://arxiv.org/abs/1905.11946`

Team, K. (n.d.). *Keras documentation: Keras applications.* Retrieved from `https://keras.io/api/applications/`

Tf.keras.metrics.auc; tensorflow core v2.8.0. (n.d.). Retrieved from `https://www.tensorflow.org/api_docs/python/tf/keras/metrics/AUC`

Tschandl, P., Codella, N., Halpern, A., Puig, S., Apalla, Z., Rinner, C., ... Kittler, H. (2020, 08). Human–computer collaboration for skin cancer recognition. *Nature Medicine*, *26*. doi: 10.1038/s41591-020-0942-0

Tschandl, P., Rosendahl, C., Akay, B. N., Argenziano, G., Blum, A., Braun, R. P., ... Kittler, H. (2019, 01). Expert-Level Diagnosis of Nonpigmented Skin Cancer by Combined Convolutional Neural Networks. *JAMA Dermatology*, *155*(1), 58-65. Retrieved from https://doi.org/10.1001/jamadermatol.2018.4378 doi: 10.1001/jamadermatol.2018.4378

Tschandl, P., Rosendahl, C., & Kittler, H. (2018, 08). The ham10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, *5*. doi: 10.1038/sdata.2018.161

Welcome to the isic challenge. (n.d.). Retrieved from https://challenge.isic-archive.com/

Yixing, F. (2020). *Keras documentation: Image classification via fine-tuning with efficientnet.* Retrieved from https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

Zaki, M. J., & Meira, W., Jr. (2020). *Data mining and machine learning: Fundamental concepts and algorithms* (2nd ed.). Cambridge University Press. doi: 10.1017/9781108564175

Zalaudek, I., Argenziano, G., Soyer, H. P., Corona, R., Sera, F., Blum, A., ... GROUP, D. W. (2006). Three-point checklist of dermoscopy: an open internet study. *British journal of dermatology (1951)*, *154*(3), 431-437.

11 Appendix

11.1 Efficient Net B3 architecture

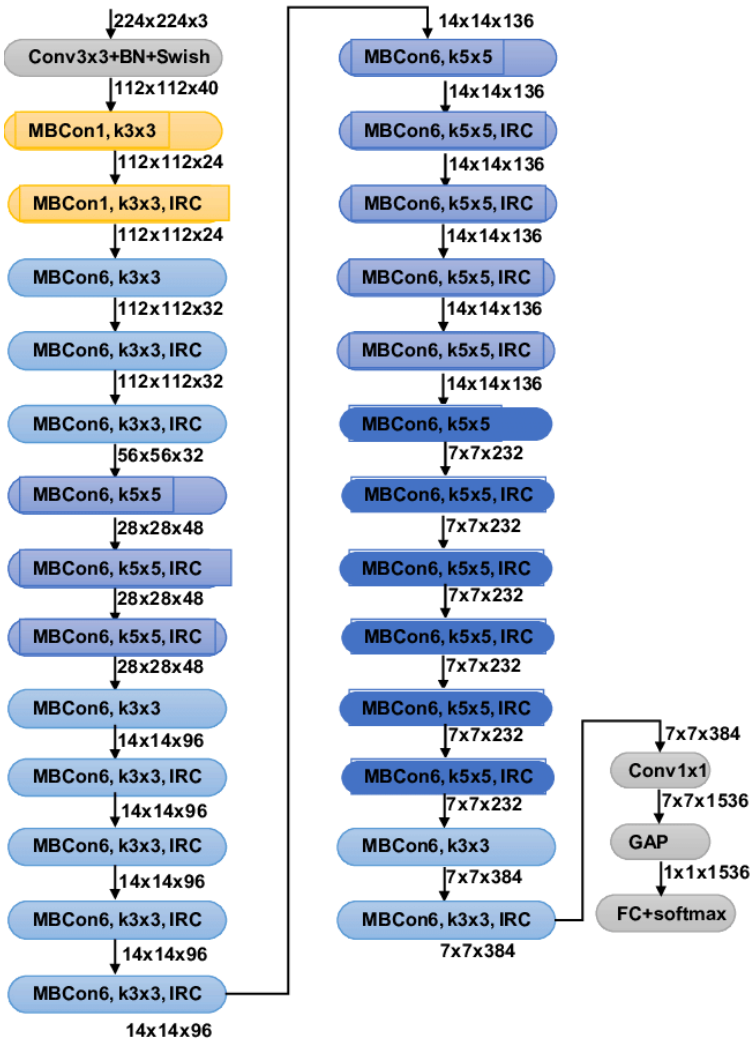

Figure 10: Overall structure of the EfficientNetB3 architecture. Each block is subdivided into segments, one example of which is figure 11. Copied from Alhichri et al. (2021)

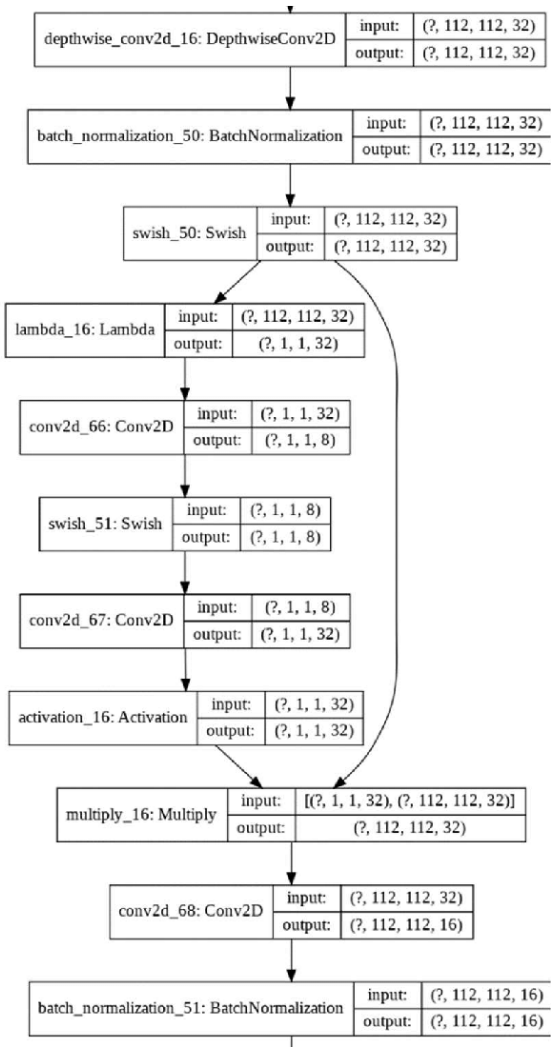

Figure 11: Block MBConv1 from the EfficientNetB3 model architecture. Copied from Alhichri et al. (2021)

11.2 Majority vote hybrid algorithm with perfect CNN

This section is a theoretical discussion on how the majority vote hybrid algorithm would perform in a situation where the CNN model has perfect performance, i.e. $TPR = 1$ when $FPR = 0$.

Let g_1 and g_2 represent two randomly selected human answers from a population and let g_{alg} represent the answer resulting from a binary encoding of the prediction probability from a computer model with perfect performance. Further let $\boldsymbol{G}$ represent the majority answer. Obtaining a majority vote can be done in a few scenarios:

$$g_1 = g_{alg} \longrightarrow g_1 = \boldsymbol{G} \tag{27}$$

$$g_1 \neq g_{alg} \longrightarrow g_2 = \boldsymbol{G} \tag{28}$$

In equation 27, the majority vote is correct since g_{alg} is always correct. In equation 28, g_1 is incorrect and the majority vote may be correct or incorrect depending on the relationship between g_1 and g_2. Since the performance of g_2 can be expected to be the average human performance, this scenario will perform equal to the human baseline. This means the performance of this approach will be somewhere between the human baseline and perfect performance.
The performance of this approach can be taken one step further. We know that taking a majority vote on human opinions is superior to the individual. This means the human answers are not perfectly correlated. Consider then situation 28. The correctness of the answer will depend on the relationship between g_1 and g_2:

$$g_1 = g_2 \longrightarrow g_1 = \boldsymbol{G} \tag{29}$$

$$g_2 \neq g_2 \longrightarrow g_1 \neq \boldsymbol{G} \tag{30}$$

The first instance gives an incorrect answer as g_1 is incorrect and the second answer gives a correct answer as g_1 is incorrect. Since g_1 and g_2 are not perfectly correlated, 30 will be relevant and I have showed that this ensemble is guaranteed to outperform human the human baseline.

(31)

11.3 Learning rate schedule

Algorithm 4: Learning rate schedule

input : Batch size, epoch number

output: Learning rate

$lr\_start = 0.000005$

$lr\_max = 0.00000125 * batch\_size$

$lr\_min = 0.000001$

$lr\_ramp\_ep = 5$

$lr\_decay = 0.8$

if $epoch < lr\_ramp\_ep$ **then**

  $lr = \frac{(lr\_max - lr\_start)}{lr\_ramp\_ep} * epoch + lr\_start$

else

  $lr = (lr\_max - lr <_m in) * lr\_decay^{epoch + lr\_ramp\_ep} + lr\_min$

11.4 Experimention with values of s

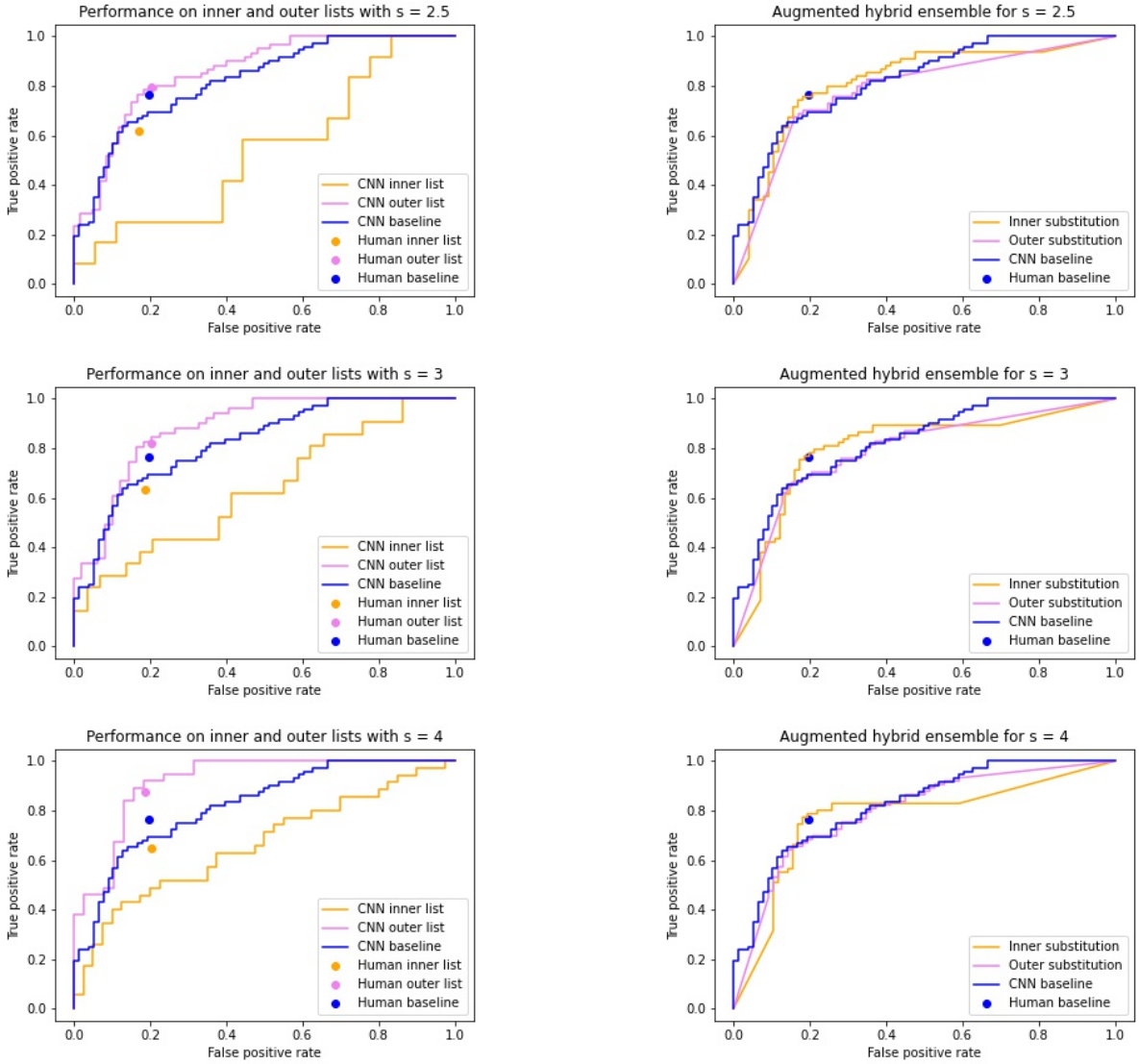

Figure 12: Baseline and hybrid majority algorithm performances for $s = [2.5, 3, 4]$. The yellow lines and dots represent the inner list, the yellow dot and line represents the outer list, and the blue line and dot represent the full data set. *Left:* The baseline performances for humans and CNN. *Right:* The augmented hybrid algorithm performances

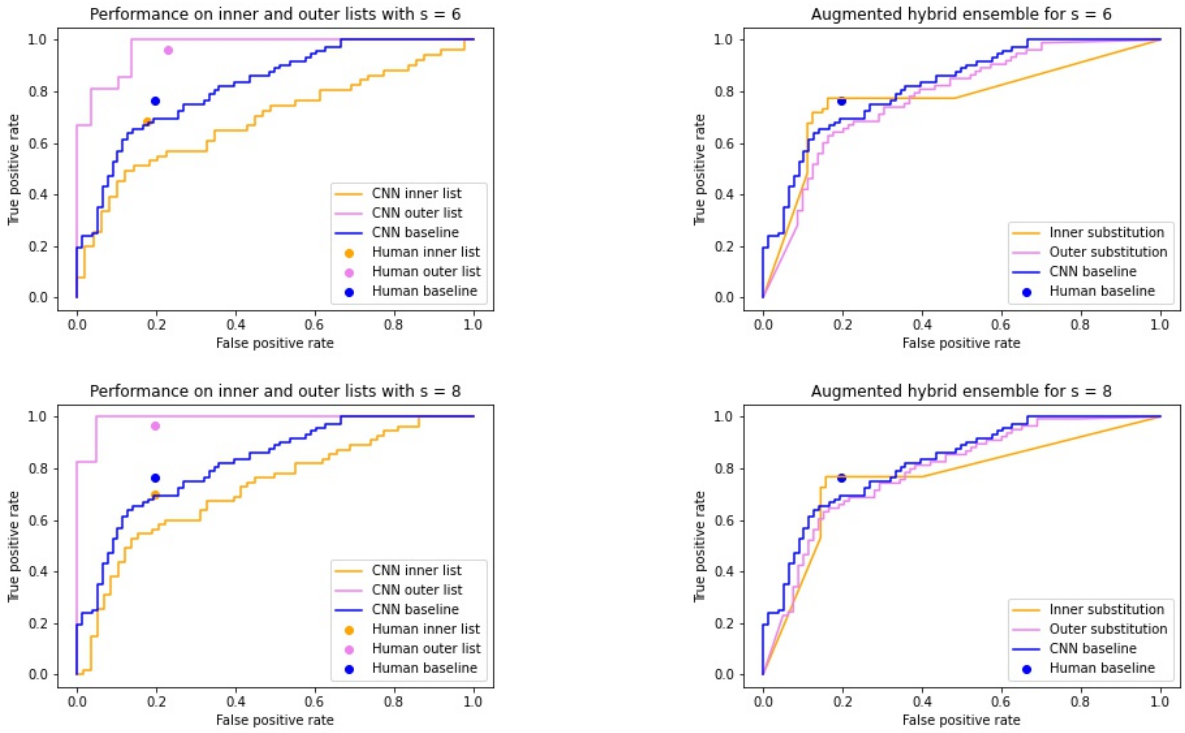

Figure 12: Baseline and hybrid majority algorithm performances for $s = [6, 8]$. The yellow lines and dots represent the inner list, the yellow dot and line represents the outer list, and the blue line and dot represent the full data set. *Left:* The baseline performances for humans and CNN. *Right:* The augmented hybrid algorithm performances