TRANSFER LEARNING IN CNNS WITH MEDICAL IMAGES

A Non-Thesis Project
presented in partial fulfillment of requirements
for the degree of Master of Science
in the Department of Computer and Information Science
The University of Mississippi

by

Dhruvin Patel

May 2019

ABSTRACT

Machine learning and data mining have been used in various real-world applications. This is due to the increase of available data people can use to train from. With the rise of data collection, machine learning has achieved meaningful advances in classification, regression, and clustering. However, many machine learning methods work well under a common premise: The training and test data are drawn from the same domain. In real-world scenarios, this is not always the case. There are times where gathering and processing data is expensive and difficult to collect. Also, when the amount of data is not significant to learn from, there needs to be a way to transfer knowledge from already previously learned domain. Which leads me to question, can I take pre-trained Convolutional Neural Networks (CNNs) features and apply it to a distant, unrelated task like medical image classification and get good performance? My findings suggests that taking a large-trained generalized CNN's features and fine-tuning to a new task has a significant effect in performance.

DEDICATION

I would like to dedicate this thesis to all my family, friends, and advisors who have inspire me to achieve my goals. Without their unwavering support, I would not be the person I am today. Thank you for all your support along the way. Lastly, I want to dedicate this thesis to my grandfather. I did not get to spend as much time as I hoped with him, but I knew that he always loved me. May he rest in peace.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

With advancements in neural networks and machine learning, the problems these topics can solve have become endless. Neural networks are being used in a wide variety of problems like stock market prediction, healthcare, object recognition, and many more. As neural networks continue to solve more problems, we should explore how we can reuse what these models have learned from their original task.

However, traditional machine learning algorithm's relied on the assumption that the input feature space and the data distribution are the same to get a well trained learned function. When data distribution and input space are different, the results of the predictive learner can be diminished [12]. In real-world applications, data for a model can change causing the performance to decrease. This leads to having to retrain and and collecting new data. It is expensive and sometimes impossible to re-collect the required training data to rebuild models [9]. This is why there is a need to create a high-performance learner for a target domain from a related source domain. These goals are the motivation for transfer learning.

Transfer learning goal is to improve learning from one domain by transferring information or knowledge from a related domain. The idea for transfer learning comes from how humans can apply related knowledge to a topic or domain. People can easily apply what they know from one domain to another related domain. For example, say for instance two student want to learn how to code in Python. One student has no prior experience in coding in any language, and the other student has learned Java or some other programming language. The student with a background in coding in another language would be able to learn

Python more efficiently, then the student who does not have any programming experience. This is because the student can transfer his knowledge in programming from one language to another. Using previous experience learned from one domain and to apply it to other, this is the idea behind transfer learning.

One example in the domain of machine learning is using pre-trained weights in image classification with CNNs. CNNs essentially take images and apply filters to extract information and train the model's weights to classify a image. For the example, say one model was trained on classifying images of fruits or vegetables, and the other model is to classify pictures of oranges or apples. The first model has numerous training examples, while the second does not have as many. If a classifier could take the first model's features and reuse it for the second task, then the model could transfer what it has learned from fruit and vegetables to detects either an orange or apple better than training on small amount of data.

Convolutional Neural Networks have had great success in image classification over the years. With the breakthrough of ImageNet [3] classifier reported by Krizhevsky et al. [7], they were able to train a large, deep convolutional neural network on a challenging dataset with 1000 classes and 1.2 million images to record breaking performance. A few years later, Donahue et al. [4] showed that a network trained on ImageNet classification was an effective blackbox feature extractor.

In this project, its goal was to experiment using two tasks that seem to be unrelated to the source. Can transferring some knowledge from the source task benefit the target tasks? Particularly, the project will show that it can take a large, deep CNN classifier trained on ImageNet and apply its features to a distant unrelated task and get better performance than training from random initialize weights in training. For the target tasks, the project explores classification on medical images. In particular, the project will use a recent dataset in skin cancer detection, HAM10000 MINST dataset [11] and another dataset of chest X-rays images for Pneumonia detection [6] for the target tasks.

CHAPTER 2

BELIEF HISTORY OF TRANSFER LEARNING

Transfer learning is not a new term in machine learning, but it has gone by many different names such as learning to learn, life-long learning, knowledge transfer, inductive transfer, multi-task learning, knowledge consolidation, context-sensitive learning, knowledge-based inductive bias, meta learning, and incremental/cumulative learning [9]. Transfer learning was first introduced in NIPS-95 workshop as "Learning to Learn", which focused on the need for lifelong machine-learning methods that retain and reuse previously learning knowledge [9]. Transfer learning is closely related to multi-task learning. Multi-task learning is learning multiple tasks at one time. The difference between multi-task learning and transfer learning is that the multi-task learns multiple task at one time, transfer learning does not.

The Board of Agency Announcement (BAA) of Defense Advanced Research Projects Agency (DARPA)'s Information Processing Technology Office (IPTO) assigned transfer learning as the ability of a system to recognize and apply knowledge and skills learned in previous task to novel tasks in 2005 [9]. From their definition, transfer learning aims to extract the knowledge from one or more source task and applies the knowledge to a target task. Today, transfer learning appears in data mining and machine learning applications.

CHAPTER 3

SYMBOLS AND NOTATIONS

In this chapter, it will formally define the notation and definition for transfer learning. The notations and definitions match those from the survey papers from [12] and [9]. Table 3.1 from [12] paper was included for a reference to the symbols and notations.

Given a domain D, it contains two parts: a feature space χ and a marginal probability distribution $P(X)$, where X = $\{x_1, ..., x_n\} \in \chi$. An example in machine learning in object classification where each pixel value is taken as feature, then χ is the space of all pixel values, x_i is the i^{th} pixel value in the image corresponding to some object, and X is a particular learning sample [9]. If two domains are different, then they may have different feature space or different marginal probability distribution [9].

Each domain D has a task T, which also has two parts: a label space Y and a predictive function $f(\cdot)$. $f(\cdot)$ is learned from the feature vector and label pairs $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in Y$. The function $f(\cdot)$ predicts the corresponding label, $f(x)$, of a new instance x. In the object classification, the example label outputs are Y, and $f(x)$ is the learned function that predicts the output label. From the above definitions, a domain is $D = \{\chi, P(X)\}$, and a task is $T = \{Y, f(\cdot)\}$.

Table 3.1. Symbols and Notations

Notation	Description	Notation	Description
χ	Input feature space	$P(X)$	Marginal distribution
Y	Label space	$P(Y\|X)$	Conditional distribution
T	Predictive learning task	$P(Y)$	Label distribution
Subscript S	Denotes source	D_S	Source domain data
Subscript T	Denotes target	D_T	Target domain data

Source domain is defined as D_s where $D_s = \{(x_{S1}, y_{S1})..., (x_{Sn}, y_{Sn})\}$. $x_{Si} \in \chi_S$ is the i^{th} data instance D_S, and $y_{Si} \in Y_S$ is the corresponding class label for x_{Si}. Also similarly, D_T is defined as the target domain task where $D_T = \{(x_{T1}, y_{T1})..., (x_{Tn}, y_{Tn})\}$, where $x_{Ti} \in \chi_T$ is the i^{th} data instance of D_T, and $y_{Ti} \in Y_T$ is the corresponding class label for x_{Ti}. Lastly, the source task is notated as T_S, the task task as T_T, the source predictive function as $f_s(\cdot)$, and the target predictive function as $f_T(\cdot)$. In most cases in transfer learning, there is one source domain D_S, and one target domain, D_T.

This leads to a formal definition of transfer learning provided by [9]. Given a source domain D_S and learning task T_S, a target domain D_T and learning T_T, transfer learning seeks to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.

From the definition above, a domain is $D = \{\chi, P(X)\}$. Thus, when $D_S \neq D_T$, that implies that either $\chi_S \neq \chi_T$ or $P_s(X) \neq P_T(X)$. Given the example previously in object classification that would mean that either the features are different between the source image set and the target image set or the marginal distribution are different.

Likewise, a task is $T = \{Y, P(Y|X)\}$. Thus, from the above definition, if $T_S \neq T_T$, then this implies that $Y_S \neq Y_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$. When $D_S = D_T$ and $T_S = T_T$, then the problem become a traditional machine learning problem. Therefore, when domains are different there are two possibilities: (1) the feature spaces between the domains are different, or (2) the marginal probability distribution is different but the feature spaces are the same. From the example of object classification, (1) would be if one image is in black and white while the other image is in color so the features would be different for each domain, and (2) if the two images set for the domain were on the same features but the image labels were different.

On the other hand, if the domain for two tasks are the same, but the task are different, there are two possibilities as well: (1) the label spaces are different between the source and task domain, or (2) the conditional probability distributions between the domains are different but the label space is the same. From the object classification example, case (1) is when the source domain has binary classes and the target domain has more than 2 classes for outputs, and case (2) is when the source and task images are very unbalanced in terms of the user-defined classes.

CHAPTER 4

RELATED WORK

There have been a number of papers that have studied transfer learning in CNNs. Yosinski et al. [13] showed a way to quantify the transferability of features for each layer of a network. Their method showed a layers generality or specificity. Yosinkski et al. [13] also found that two distinct issues that affect how transferability is affected: splitting the network in the middle and higher layers specialization to the original task.

A few papers have researched various factors that affect pre-training and fine-turning in CNNs. Agrawal et al. [1] analyzed how to use a data-rich dataset like ImageNet to initialize a CNN parameters and train on a small dataset as a feature extractor or to continue to train, which is called fine-tuning. Agrawal et al. [1] and Yosinski et al. [13] examined whether pre-training should be stopped early to prevent over-fitting, and what layers should be used for transfer learning. On the other hand, Zhizhong et al. [8] proposed a way to fine-tuning a model with a new task without "forgetting" the old ones.

Huh et al. [5] questioned what makes ImageNet good for transfer learning. They found the number of classes or the number of images used when training has only a modest effect on the transfer task performance. This is going against common assumptions that the number of label images correlates to the success of getting highly generalizable deep features.

Outside of transfer learning in CNNs Pan et al. [9] and Weiss et al. [12] both gave a comprehensive review of transfer learning in machine learning but with more focus on data mining. They formally defined transfer learning and surveyed the different categorizes and techniques on transfer learning. While CNNs primarily use supervised data, [9], [12], and [2] explored transfer learning in unsupervised learning.

CHAPTER 5

DATASETS

The datasets in the experiments were chosen from Kaggle, an open community of data scientists where datasets are accessible to the public. The project was to test if using a generalized unrelated dataset can improve the performance for a specific medical image dataset. Therefore, the generalized unrelated dataset will be ImageNet. The two specific medical image datasets will be a recently published dataset in skin cancer detection, HAM10000 MINST dataset [11], and the other dataset is Pneumonia detection in chest X-Rays [6].

5.1 ImageNet

ImageNet is a large visual dataset with over 14 million images. The dataset has over 20,000 categories. The categories contain a wide variety of objects such as cats, dogs, chairs, food, etc. [7]. This makes the dataset generalized but still unrelated to the medical images. Because of resource limits, the project does not train the models from ImageNet from scratch. The models used Keras's pretrained ImageNet weights implementation.

5.2 HAM10000 MINST

HAM10000 consists of 10,015 dermatoscopic images of common pigmented skin lesions. The original image sizes are (600, 450), which were resized to (224,224). The dataset has seven classes in which each image belongs to one class. The different classes include a compilation of all important diagnostic categories in pigmented lesions: Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses,

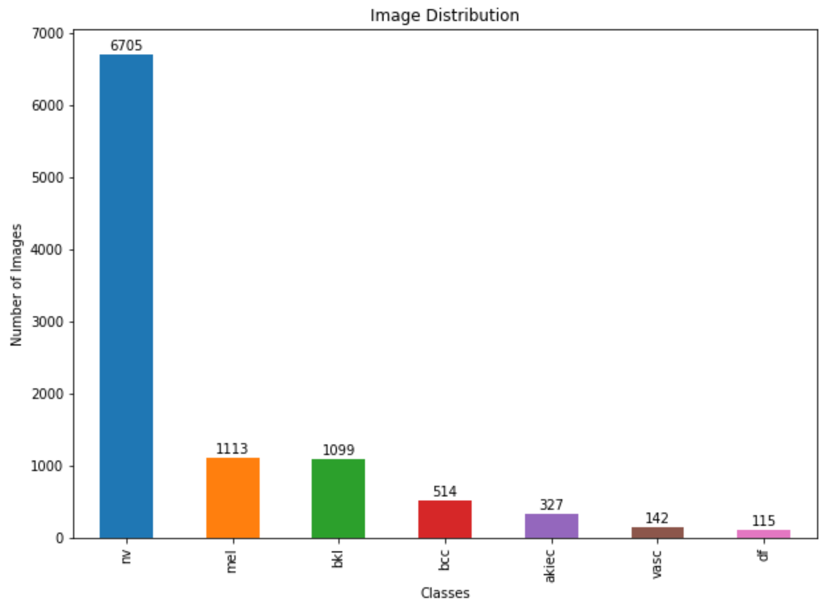


Figure 5.1. Image Distribution for Skin Cancer Dataset

bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc). Out the 7 classes, the most malignant classes are akiec, bcc, and mel and the benign classes are bkl, df, nv, and vasc. For more information about the different types of skin lesions refer to [11].

One challenge this dataset presents is the unbalanced nature of the classes. In Figure 5.1, the bar chart displays the number of images per class. Most of the images in the dataset belong to the class nv. Approximately, 80 percent of the images belong to the benign classes, and some classes like akiec, vasc, and df make up less than 3 percent of the total dataset. These issues present a challenge when trying to build a classifier to detect the classes. One assumption in computer vision is that the more images a classes or dataset has the better the learned predictor will be. Therefore, getting a desirable predictor will be a challenge.

5.3 Chest X-Ray Dataset

The other dataset used in the project is from [6], which consist of 5,856 chest x-rays of which 1,583 are normal and 4,273 have Pneumonia. Figure 5.2 gives a visualization of

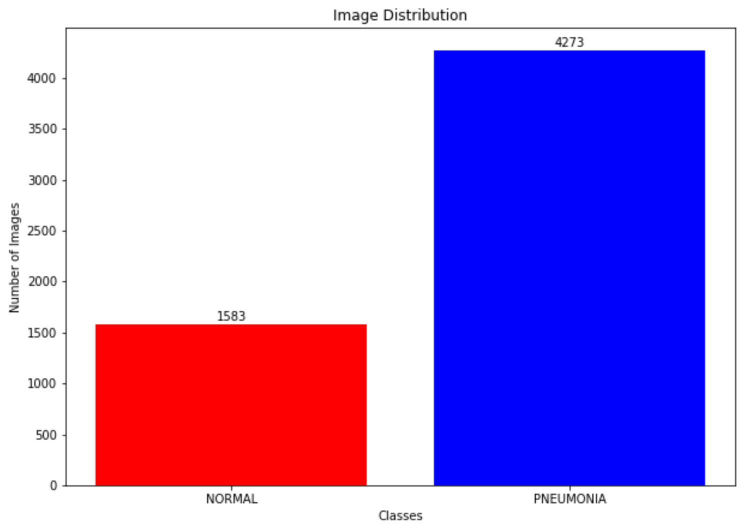


Figure 5.2. Image Distribution for X-Rays Dataset

the number of images per class. The original image sizes are (2090, 1858), which similarly to HAM10000 were resized to (224,224). Since this dataset only has 2 classes and each class has more than 27 percent of images, the learned predictor will have better performance with this dataset then HAM10000. In chapter 7, the results will show how the datasets can be applied to pretrained weights.

CHAPTER 6

EXPERIMENT SETUP

The experiments are broken down into two parts: baseline models and fine-tuned models. The baseline models will be a CNN, decision tree, and random forest classification. The transferred models will be a CNN with ImageNet pre-trained weights and fine-tuned to the skin cancer dataset and x-ray dataset. Pre-training is a mechanism where a model's CNN parameters are initialized using the task of ImageNet classification. Moreover, fine-tuning is a process where a model adapts the pre-trained CNN weights to continuously train on a target dataset.

6.1 Decision Tree and Random Forest

For the decision and random forest classification, the datasets had a 80/20 split where 80 percent of the images went to training and 20 percent to the test. A decision tree is like a choose your own adventure. Based on the path chosen, the type of adventure is splits into branches like a tree. In this project based on the pixel values, the model picks the best pixel value to split into paths until it can classify the image. Random forest builds multiple decision trees and merges them together. The model randomly chooses the best feature from a random subset of data, hence called random forest. The decision tree and random forest were implemented using ski-learn's python library. For the features of the classifier, the model was tested using the pixel values, normalized pixel values, and rescaled pixel values. However, using the pixel values for the features performed the best. Thus in the results, the features for the classifier were using the original pixel values.

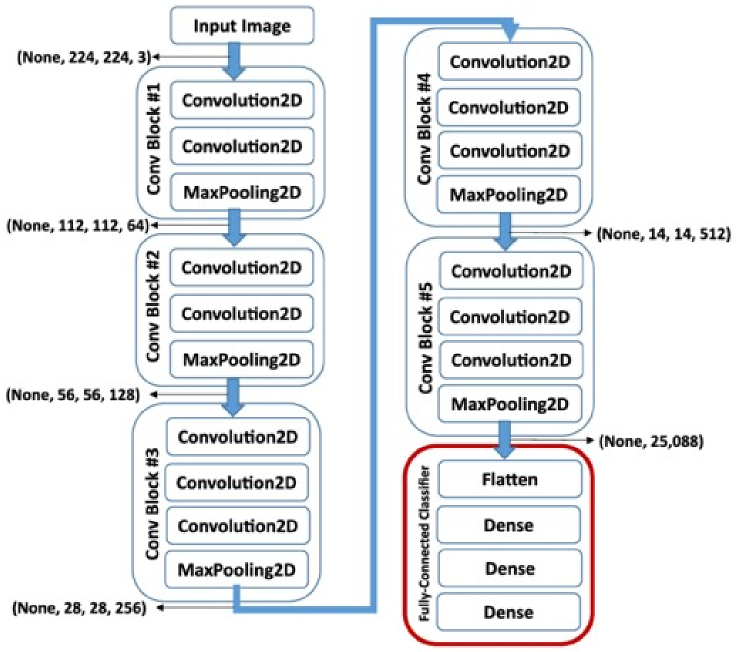


Figure 6.1. VGG-16 Model

6.2 CNN

For these experiments, the datasets were split into 60/20/20 split. Therefore, 60 percent of the images were used for train, 20 percent for validation, and 20 percent for testing. Moreover, since in the HAM10000 dataset a few of the classes had few images to train from, the images were augmented when training the CNN. Since the model is using ImageNet weights to initialize the CNN, the images had to be normalized in the same way VGG-16 was trained using ImageNet. Therefore, the images were rescaled so that each pixel value was between -1 to 1. To stop over-fitting, the models used an early stop that would stop training when the validation accuracy does not improve after 5 epochs.

All of the experiments using a CNN used a Keras implementation of VGG-16, which was proposed by Simonyan et al. [10]. In Figure 6.1, the image gives a view of the schematic of the VGG-16. VGG16 model consist of a 5 convolution blocks and one fully connected block. The convolution blocks have 2 convolutional layers for the first two blocks and 3 convolutional layers for the other 3 blocks and a max pool layer at the end of each block. Since the fully connected layers are generally assumed to capture information applicable to the source task, all the models were set to random weights for the fully-connected layers. Also, in the model the fully connected layers were replaced with 1 flatten layer, 2 dense layers, and 2 dropout layers between the dense layers and one last dense layer.

Using the HAM10000 dataset, the goal was to test if ImageNet weights can benefit performance and if fine-tuning specific block of the model makes a significant difference. Therefore, the results will show the F1-scores of fine-tuning with all layers trainable, first and last blocks trainable, last block trainable, and first block trainable. Refer back to the Figure 6.1 for which layers the experiments froze during training. Using the X-ray dataset, the experiment tests if ImageNet weights are better than random weights or weights from HAM10000 training. The experiments followed similar training to [1] when using ImageNet weights for the experiment. Some changes in the experiments were to use stochastic gradient descent instead of Adam and setting the learning rate to 1e-4. Using the x-ray dataset, the experiment will compare using wights from ImageNet and weights from the model trained on the HAM10000.

6.3 Development Environment

- Windows 10
- GPU: GTX 1080 (supports Cuda)
- Cuda 9.0
- Python 3.6
- Keras 2.2.4
- Numpy 1.15.4
- Mathplotlib 3.0.2

CHAPTER 7

RESULTS

The results of the experiments are split into two sections for each dataset.

7.1 HAM10000 Results

Table 7.1 displays the F1-scores for all the classes, micro average, and macro average using HAM10000 dataset. Out of the baseline models, the CNN not using ImageNet weights was to achieve 72 percent micro average; however the macro average of the model is 35 percent. This is primarily due to the classes with lower amount of images causing the macro average to decrease. All of the models were able to predict the class nv well, but this is because more than 60 percent of the dataset belonged to the class.

The more interesting result from the table is that using the ImageNet weights to initialize the model was a sufficient boost to performance in all the classes and accuracy. The macro average nearly doubles and the micro average is 9 percent higher than the model with random weights.

To analyze the benefits of the ImageNet pre-trained weights has on the network, Figure 7.1 shows the training accuracy from a model trained without ImageNet weights and one model with ImageNet weights. The red dashed line is the model using ImageNet weights. The blue solid line is the model using random weights. From the figure, it can be concluded that there are three significant findings.

- Using pre-trained weights, the model has a **higher slope** when training.
- When training is complete, the model using pre-trained weights has a **higher accuracy**.

	Baseline Models			CNN Models using ImageNet Weights			
	Decision Tree	Random Forest	CNN Random Weights	All Layers Trainable	Last Block Trainable	First Block Trainable	First and Last Blocks Trainable
akiec	.09	.25	.31	.36	**.27**	.00	.36
bcc	.20	.38	.41	.64	.59	.00	.64
bkl	.26	.40	.37	.64	.58	.00	.60
df	.00	.00	.00	.27	.00	.00	.07
mel	.19	.22	.34	.58	.54	.01	.53
nv	.76	.84	.88	.92	.91	.80	.91
vasc	.06	.00	.17	.78	.50	.00	.50
micro avg	.57	.70	.72	.81	**.79**	.66	**.79**
macro avg	.23	.30	**.35**	**.60**	**.48**	**.11**	**.52**

Table 7.1. F1-scores of baseline models and CNNs models on HAM10000.

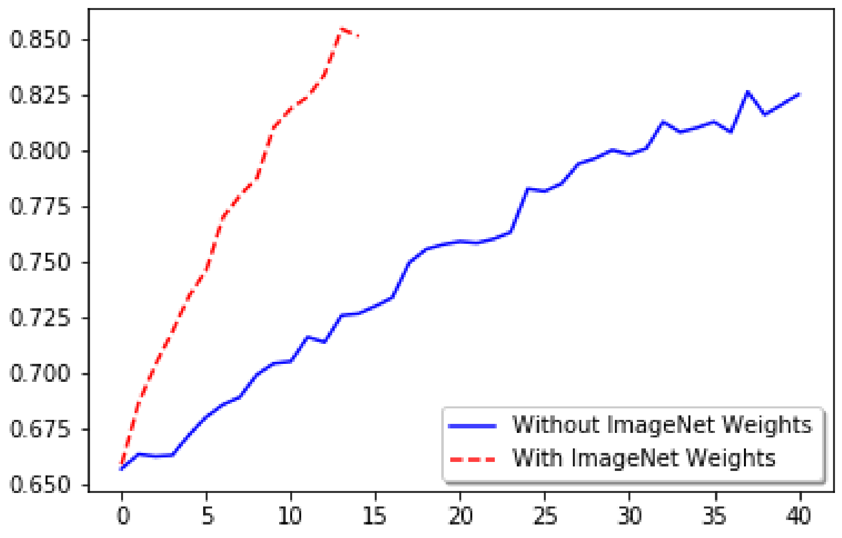


Figure 7.1. Line graph of training accuracies of two models: Using ImageNet Weights and Without ImageNet Weights

- The model using ImageNet weights was able to **finish training in few number of epochs** compared to random weights.

This suggest that ImageNet weights are highly generalized, which make it ideal to start training from these weights rather than random weights. Therefore, using ImageNet weights are an effective feature extractor that can be used on distant, unrelated tasks from ImageNet.

In addition to the benefit of using the ImageNet weights, the other tests explored training on different layers to see which layers benefit the most from using the ImageNet weights. From Table 7.1, it can be conclude that training on the later layers results in a better performance boost. This make sense since the later layers in a CNN are task specific, and the beginning layers are more generalized. When training on just the beginning layers, the model was only able to achieve 11 percent accuracy. This lead the model to predict all the images in one class, nv. This shows that the later convolutional layers are not beneficial when the model is not fine-tuned to the target task.

While training with the last layers, the model had similar results to training on the first and last layers. Only a few classes benefited from unfreezing the beginning layers. However, when the model was trained on all the layers it was able to get the best performance.

7.2 X-ray Dataset

Using the X-ray Dataset, the experiments differ from the HAM10000 dataset because it tests using the model weights from HAM10000 trained model, but the experiments also include the baseline models and a model using ImageNet weights. In Table 7.2, it displays the F1-scores of Normal and Pneumonia classes and micro and macro averages of the different classification models. The baseline of the model performed sufficiently better compared to the HAM10000 models. Random forest only performs sightly worse than the CNN with random weights. With random weights the normal class had a 1 percent increase compared to the random forest.

	Baseline Models			Transferred Models	
	Decision Tree	Random Forest	CNN Random Weights	CNN ImageNet Weights	CNN HAM10000 Weights
Normal	.75	.88	.89	**.91**	.90
Pneumonia	.91	.96	.96	**.97**	.96
micro avg	.87	.94	.94	**.95**	.94
macro avg	.83	.92	**.92**	**.94**	**.93**

Table 7.2. F1-scores of baseline models and CNNs models on X-ray dataset.

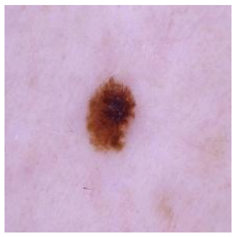
Out of the CNNs models the most striking result was that using ImageNet weights still outperformed all the models including the CNN using HAM10000 weights. Using ImageNet weights, increase the accuracy in both classes therefore boosting the micro and macro averages. This could suggest that the HAM10000 weights are more specific to the original task and that the ImageNet weights are more generalized. On the other hand, using the HAM10000 weights, it was able to outperform the random weights and the baseline models. Similarly to CNN with random weights compared to random forest, the CNN using HAM10000 weights was able to increase the accuracy in the normal class by 1 percent which lead to the macro avg increase of 1 percent.

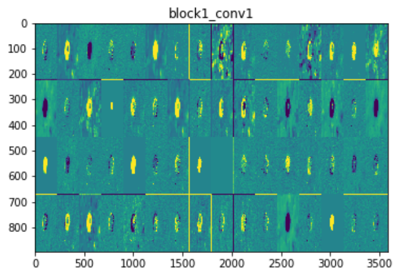
CHAPTER 8

DISCUSSION

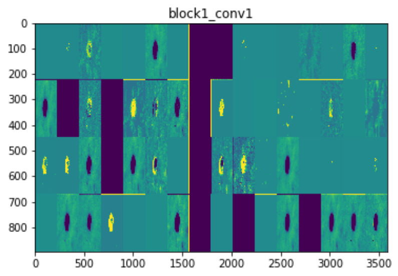
In the results in chapter 7, it shows the positive effects of using fine-tuned pre-trained weights on medical images. In particular, it displays that by leveraging a large trained model on ImageNet, the target task can use the weights to learn features that are unrelated to the source task. Notably, pre-trained ImageNet weights showed to have a significant boost in performance compared to random initialization. This suggests that ImageNet weights are a favorable starting parameters than random initialization. To visualize how ImageNet affects the model, Figure 8.1 presents a side by side of a skin lesion image (a) and the image activation from the first convolutional layer of the models with ImageNet weights (b) and with random weights (c). From the Figure 8.1, the model with using ImageNet has more of the lesion highlighted in the activation than the activation with random weights. Thus, the model with ImageNet weights are extracting more information from the images than the model with random weights.

From the experiment with the HAM10000 dataset, it can be concluded that there are three benefits with using pre-trained weights: faster training, steeper training slope, and


(a) Original Image



(b) Activation with ImageNet



(c) Activation with Random Weights

Figure 8.1. Visualize of intermediate activation from first convolutional layer

higher accuracy. Also, it found that training on the later layers has a greater impact than the beginning layers. This makes sense because later layers in the CNNs are "more specific" to the task, while the early layers are generalized.

In the X-ray experiment, it was showed that the ImageNet weights had the best accuracy beating random and weights from HAM10000 model. Interestingly, this suggest that the weights from HAM10000 were not as generalized as ImageNet weights. However, the HAM10000 weights did achieve better accuracy than random weights, this is because the weights were first ImageNet weights that were fine-tuned to the skin cancer images. Thus, the weights still retained some of the features of the ImageNet weights.

Although the experiments have success in classifying medical images, there are some downsides. One downside is the model itself. Since the experiment was using VGG16 as the model, the recommend image size is (224,224). Therefore, if the image size is small in a dataset, then there is a chance the pre-trained weights will not benefit the performance of the target task. Another possibly downside is that both dataset had plentiful images to train from, and if the target dataset were smaller there could be negative transfer of the features.

CHAPTER 9

CONCLUSION

All in all, from the experiments it can be concluded that a target task can benefit from a source task even if the source task is seemly unrelated to the target task. Using weights that were trained on ImageNet, the model was able to extract the features and apply them to medical images to achieve higher accuracy. Transferring the weights showed to be better than training using standard random weights. While the experiments showed to have positive knowledge transferred, there are still downsides. When using ImageNet weights, the model has to be pretrained on ImageNet and has to follow the same training setup. This is not practical for all image datasets. Overall, the project presents that by taking a well generalized source task it can apply to unrelated specific target task and have impressive performance.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Pulkit Agrawal, Ross B. Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. *CoRR*, abs/1407.1610, 2014.

[2] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1:2012, 2012.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[4] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[5] Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016.

[6] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

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

[8] Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

[9] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[11] Philipp Tschandl. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions, 2018.

[12] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3, 2016.

[13] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.