Final Year Project Report

Full Unit - Interim Report

Comparison of Image Classification Models with Transfer Learning

James Arnott

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Li Zhang

Department of Computer Science
Royal Holloway, University of London

November 24, 2022

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: James Arnott

Date of Submission:

Signature:

Table of Contents

Abstract

Transfer learning is a common, efficient method for training deep learning models. It involves using a pre-trained model to extract features from a dataset, and then training a new model on top of the extracted features. This report compares a variety of different image classification models with transfer learning, and evaluates their performance on a dataset of images of flower species and MRI scans of Alzheimer's patients brains.

Project Specification

Your project specification goes here.

Chapter 1: **Introduction**

1.1 What transfer learning is

Transfer learning is a common, efficient method for training deep learning models. There are many different ways to implement transfer learning and here I hope to compare the performance of a variety of different methods with different pre-trained models.

The applications for transfer learning are vast, and it is a common method for training deep learning models, This is because it is a very efficient method for training models as most of the work is done by the pre-trained model. The pre-trained model is used to extract features from the dataset, and then a new model is trained on top of the extracted features. The pre-trained model is usually frozen so that it does not change during training, and only the new model is trained.

After the new model is trained on top, a process known as fine-tuning can be used to further improve the performance of the model. Fine-tuning involves training the new model on the dataset, but with a lower learning rate than the initial training. This is also where the model is usually unfrozen, so that the pre-trained model can be trained as well, which helps to improve the overall performance of the model.

1.2 The Models I will be comparing

The models I have chosen to compare are the MobileNetv3Small[4], MobileNetv3Large[4], ResNet50[3], ResNet10[3], ResNet152[3], and InceptionV3[7] models. These models are all pre-trained on the ImageNet dataset, which contains 1.2 million images of 1000 different classes.

1.3 The Datasets I will be using

The datasets I have chosen to use for this project are the Oxford 102 Flower Dataset[6] and the Alzheimer's MRI Dataset[1].

The choice of datasets is based on the fact that they are both image classification datasets they are both relatively small in classes and the they have a variety classes ranging in difficulty, which will allow me to compare the capabilities of the different models in a variety of different scenarios.

Chapter 2: **The way Neural Networks work**

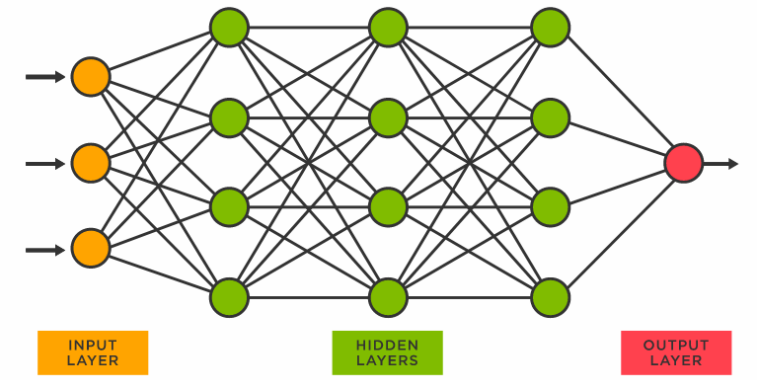

Figure 2.1: A diagram of a neural network [5]

2.1 What is a Neural Network

A neural network is a machine learning model that is inspired by the way the human brain works. It is made up of a series of layers, each layer is made up of a number of neurons. Each neuron is connected to every neuron in the next layer, and each connection has a weight associated with it. The weights are used to determine how much each neuron in the next layer is affected by the current neuron. The weights are updated during training, and the process of updating the weights is known as backpropagation.

2.2 What is a Convolutional Neural Network

A convolutional neural network (CNN) is a type of neural network that is used for image classification. It is made up of a series of convolutional layers, pooling layers, and fully connected layers. The convolutional layers are used to extract features from the image, and the pooling layers are used to reduce the size of the image. The fully connected layers are used to classify the image. They are often much more efficient than other types of neural networks, and are commonly used for image classification.

Chapter 3: **Transfer Learning**

3.1 Why use Transfer Learning

Transfer learning can enable a model to be trained much more efficiently than if it was trained from scratch, as most of the work is done by the pre-trained model. Aside from the extra time it takes to train a model from scratch, it often requires more training data, which is not always available and can be expensive to obtain. Transfer learning can often get better results than training a model from scratch, as the pre-trained model has already learned about different features from a large dataset, and the transfer learning model can then use these features to learn about the new dataset.

3.2 How to implement Transfer Learning

3.2.1 Extracting features

Feature extraction is the process of using a pre-trained model to extract features from a dataset. The pre-trained model is usually frozen so that it does not change during training, and only the new model is trained, and the new model is trained on top of the extracted features.

3.2.2 Fine-tuning

This is the process of training the new model on the dataset, but with a lower learning rate than the initial training and the model is usually unfrozen, so that the pre-trained model can be trained as well, which helps to improve the overall performance of the model. This was where I managed to get the best results for my models.

3.3 How to fine-tune a model

Fine-tuning a model is a very simple process, and can be done in just a few lines of code. The first step is to unfreeze the pre-trained model, which can be done by setting the trainable attribute of the layers to True. The next step is to set the learning rate to a lower value, which can be done by setting the learning rate attribute of the optimizer. The final step is to recompile the model, which can be done by calling the compile method of the model. Here is an example of how I fine-tuned a model using keras:

```
base_model.trainable = True
optimizer.learning_rate = 0.0001
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
```


```
base_model.trainable = True
optimizer.learning_rate = 0.0001
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
```


3.4 The different structures of the models

There are so many different ways to structure a model and they can

Chapter 4: **Evaluating the performance**

4.1 The different metrics

Accuracy is the most common metric used to evaluate the performance of a model, and is defined as the number of correct predictions divided by the total number of predictions. It is a very simple metric to understand, and is therefore often used to compare the performance of different models. There is accuracy for both the training and validation sets, and the accuracy of the validation set is used to determine how well the model generalises to new data, and a validation accuracy that is much lower than the training accuracy is a sign that the model is overfitting. However, it is not always the best metric to use, as it can be misleading in some cases.

Loss is the most common metric used to evaluate the performance of a model, it is calculated by taking the difference between the predicted value and the actual value. There are however many different types of loss functions, and the one that is used depends on the type of problem that is being solved. These could be binary cross entropy, categorical cross entropy, or mean squared error. For my project I used binary cross entropy due to it's simplicity and the fact that it is the most common loss function used for binary classification problems.

The validation loss and accuracy are used to determine how well the model generalises to new data, and a validation loss that is much higher than the training loss is a sign that the model is overfitting. For the validation set, I have assigned 20% of the data to be used.

The confusion matrix is a table that shows the number of correct and incorrect predictions for each class. It is a very useful metric to use when evaluating the performance of a model, as it can show which classes the model is having the most trouble with.

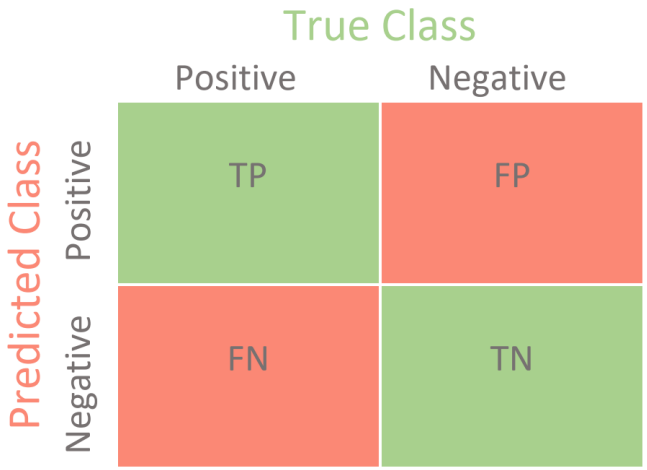

Figure 4.1: A confusion Matrix[2]

4.2 The different datasets

4.3 The different sizes and structures of the models

4.4 The different optimizers

Chapter 5: **Optimising the models**

Chapter 6: **Conclusion**

This project is a great learning experience for me, and I have learned a lot about transfer learning and how to implement it. Before I started this project, I had never used transfer learning before, and had only used pre-trained machine learning models for image classification. I have learned a lot about the different types of models, how they are structured, and how they can be used for transfer learning.

6.1 What I have learned

I have specifically learnt how complex image classification models are structured, as they are far more complex than I had initially thought.

6.2 What I will do differently

I need to spend more time researching the different structures of the models, and how the optimal ways they can be used for transfer learning.

6.3 What I need to do next

Bibliography

[1] *Alzheimer's Dataset*. URL: https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset.

[2] *Confusion Matrix*. URL: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

[3] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[4] Andrew Howard et al. "Searching for MobileNetV3". In: *CoRR* abs/1905.02244 (2019). arXiv: 1905.02244. URL: http://arxiv.org/abs/1905.02244.

[5] *Neural Network Diagram*. URL: https://www.tibco.com/reference-center/what-is-a-neural-network.

[6] *Oxford Flowers 102*. URL: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/.

[7] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.