CPSC 66 Final Report: Transfer Learning with Real-World Applications in Deep Learning

William Huang WHUANG2@SWARTHMORE.EDU
Anarsaikhan Tuvshinjargal ATUVSHI1@SWARTHMORE.EDU
Zeus Borrego ZBORREG1@SWARTHMORE.EDU

Abstract

CNNs can be trained from scratch or utilize pre-trained models to implement transfer learning. Transfer learning can dramatically improve testing accuracy compared to models trained from scratch, especially with tiny datasets. This project explores how models trained from scratch and pre-trained models compare in testing performance on classifying brain tumor scans. Using Tensorflow libraries, we found that complex pre-trained models worked exceptionally well on small datasets, although taking longer to fit. However, as training data increased, the advantage of complex pre-trained models began to diminish. Our results led to interesting discussions regarding the contexts in which we prefer specific models.

1. Introduction

Convolutional Neural Networks (CNNs) are often used in image recognition tasks. CNNs can efficiently be utilized for transfer learning, the process in which knowledge gained from solving one problem can be applied to a different but similar problem, generally providing performance improvements for smaller datasets.

We sought to explore how CNNs work and investigate the benefits of transfer learning. We began by building and optimizing our own CNN models on Kaggle's Cat vs. Dog dataset. It contained labeled pictures of cats and dogs, with the goal of our model to predict if a picture is of a cat or a dog. We picked the best-performing model and used it as our baseline for comparison. Then, using the TensorFlow and Keras libraries, we imported several pre-trained models and implemented them for transfer learning, observing the change in performance on the same dataset.

CPSC 66 Machine Learning Proceedings, Swarthmore College, Spring 2021.

Once we developed a solid foundation for building CNNs and applying transfer learning, we compared accuracy on another dataset, whose validation accuracy would have more serious implications. We went with the Brain Tumor dataset, which contained brain scans labeled with respect to the existence of a brain tumor (healthy/cancer). We trained a new CNN model from scratch on this dataset, applied transfer learning with two different pre-trained models, and compared performance between the models.

Two primary questions guided our experiments:

- Can we use CNNs that are trained to classify one specific set of images (i.e., cat or dog) and transfer that trained model to classify a separate set of images (i.e., brain tumor or no brain tumor)?
- Can we utilize transfer learning for optimization purposes like reducing the cost of resources or increasing performance on classification problems?

2. Methodology

We broke our experimentation procedure down into three main phases:

- Building and optimizing our own CNNs from scratch,
- Importing pre-trained models and implementing transfer learning,
- Training new models for a different dataset.

The basic idea of transfer learning is pretty straightforward: Train a model on a large dataset and transfer its knowledge to a smaller dataset. For our experiment, we froze the early convolutional layers of the model and only trained the last few layers, which are the layers responsible for making label predictions. CNNs can take advantage of transfer learning because convolutional layers extract general, low-level features applicable across images like edges, patterns, lines. Because of the universal, low-level features

shared between images, we can use a model trained on different image classification tasks and apply it to another image classification problem.

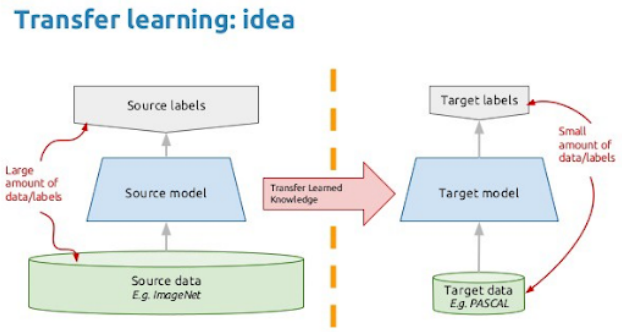

Figure 0. Idea behind Transfer Learning

3. Building and Optimizing Our Own CNNs from Scratch

We began by building our own CNN models. The architecture of a CNN is characterized by one or more blocks, each block containing one or more convolutional layers followed by a single pooling layer and an output layer at the end.

We constructed a simple initial CNN model consisting of a single block with one convolutional layer and one pooling layer. All CNNs must contain one fully connected dense layer at the end for the output. We decided that every block we constructed would only have one convolutional layer and one pooling layer for experimental consistency. Using the validation accuracy of that model as a baseline, we experimented with hyperparameter tuning to increase that baseline validation accuracy. We then built a model with two, three, four, and five blocks and found that each subsequent model provided better validation accuracy.

It became apparent that hyper-parameter optimization would be a poor allocation of time. Even if we did find some ideal set of hyper-parameters, those values would only apply to the specific dataset and potentially provide little to no generalization value. With that and the fact that each attempt at fine-tuning required approximately fifteen minutes to train the model with new values, we decided to use our five-block model as a benchmark for further comparisons.

We applied two additional techniques to improve our model: dropout regularization and batch normalization. During training, some neurons in the network may contain substantial weights, heavily influencing the output. Dropout is a regularization technique that randomly shuts off neurons in the network during each epoch to avoid such overfitting. We also used batch normalization to allow every layer to learn independently to avoid overfitting by normalizing the output of previous layers. We applied each first in isolation and then with both together. The addition of either technique improved our five-block model, but we observed the best performance when we included both.

We applied the following controls to each model for experimental consistency:

- Trained each model on ten epochs
- Each block contains one CNN layer and one pooling layer
- Doubled the number of neurons in each subsequent block starting at thirty-two
- Used rmsprop as the optimizer
- Trained on Cat vs. Dog dataset

4. Importing Pre-trained Models and Implementing Transfer Learning

Establishing our baseline lets us gauge how well a CNN model might perform. The next step in our experimentation was to test the performance of pre-trained models on the cat/dog classification task after applying transfer learning. We tested with three pre-trained models: VGG16, Inception-ResNet-v2, and Inception V3. Since pre-trained models are generally trained on extensive datasets, we expected them to perform better than the model we trained from scratch. We wanted to determine the extent to which the pre-trained models would outperform our baseline model and do a cost-benefit analysis on the tradeoff between performance and runtime.

5. Training New Models for a Different Dataset

The final portion involved the same procedures we had followed previously but with a different dataset (Brain Tumor). We built three models in this step: the first was trained from scratch using the Brain Tumor dataset, the second utilized transfer learning with the pre-trained Inception-ResNet-v2, and the third utilized transfer learning with our CNN model that had been pre-trained on the cats/dog classification problem. We expected that the pre-trained Inception-ResNet-v2 model would outperform the other two models.

We chose to create these models in particular in hopes of drawing attention to a few concepts:

- Comparing transfer learning using our pre-trained five-block CNN model with the new model trained from scratch highlights the performance impact of transfer learning
- Comparing the pre-trained Inception-ResNet-v2 model with our initial five-block CNN model highlights the performance impact of using an established pre-trained model
- Comparing the pre-trained Inception-ResNet-v2 model with the model trained from scratch highlights both the performance impact of using established pre-trained models and transfer learning

6. Experiments and Results

The Brain Tumor dataset was retrieved from Kaggle and consists of images of brain x-rays of patients with healthy brains and x-rays of patients diagnosed with brain tumors. The brain scans originally had significant variations in pixel density, so we modified the images to be 128x128 pixels for experimental consistency since we trained our original five-block CNN on 128x128 images of cats and dogs. Both our test and validation sets had 230 images. During experimentation, we altered the training set size (500, 1000, 2000, and 4000) to compare accuracies across different models as the number of training examples increased.

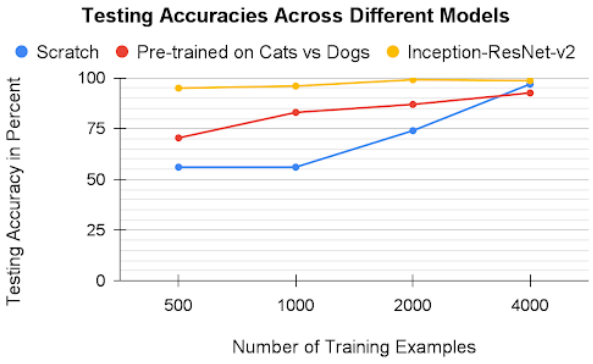

Figure 1.

To visualize the testing accuracies across different models with different dataset sizes, we compiled our results in Figure 1. We have the number of training examples on the horizontal axis, starting at 500, doubling with each increment. On the vertical axis, we have the testing accuracy in percent. The legend shows three lines, each illustrating the change in testing accuracy of a particular model as the number of training examples increases.

Immediately, we notice that regardless of which model we observe, the overall trend is that testing accuracy increases with the number of training examples. This behavior is most salient with the model trained from scratch, which went from just over 55

The Inception-ResNet-v2 model was unambiguously better than the other two, regardless of the number of training examples. Even with 500 examples, the testing accuracy was above 95

The scratch model performed the worst at 500 examples but eventually surpassed the model pre-trained on Cat vs. Dogs at 4000 examples. With 500 examples, the model pre-trained on Cats vs. Dogs outperforms the scratch model by approximately 15

It makes sense that using pre-trained models would give an initial boost in training accuracy when dealing with smaller datasets. Since pre-trained models have already been trained on other datasets, we expect them to exhibit a noticeable advantage compared to models trained from scratch, especially for smaller datasets. However, as the number of examples in the dataset increases, training from scratch catches up with the pre-trained models. These principles are especially salient when we compare the scratch model to the model pre-trained on Cats vs. Dogs: the latter exhibited an initial upper hand in testing accuracy, but that difference diminishes over time. The former eventually manages to surpass the latter in testing accuracy.

Comparing the two pre-trained models, it is apparent that Inception-ResNet-v2 has superior testing accuracy on all accounts. This result makes sense because it is a far more complex model trained on more than a million images from the ImageNet database compared to our model trained only on the Cats vs. Dogs dataset.

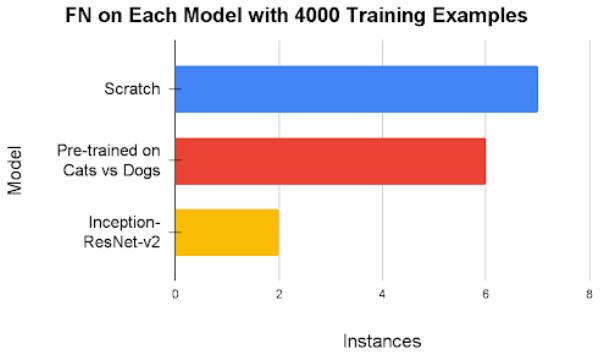

Figure 2.

For a classification task like Cats vs. Dogs, false negatives (FN) and false positives (FP) have negligible consequences. However, if the medical field were to adopt a ma-

chine learning algorithm for detecting brain tumors, getting an incorrect diagnosis would have much more dire consequences. In particular, FN would have much more significant repercussions than FP. An FP may lead to inconveniences for patients, as they would be required to undergo further procedures even though they are unnecessary. The false alarm can also lead to stress for patients and their families. However, a FN could prevent an early intervention, potentially delaying life-saving operations.

For this reason, we wanted to focus on the number of FN in each of our models. Since all three models had the highest testing accuracy with 4000 training examples, we wanted to compare the number of FN each model had, holding the number of training examples constant at 4000. We created confusion matrices for the three models of interest to observe this.

We compiled the FN on each model with 4000 training examples in Figure 2. We have the number of instances that a particular model had FN on the horizontal axis. On the vertical axis, we have the three types of models we used in this experimentation phase. Each bar depicts the frequency of FN for each model.

The Inception-ResNet-v2 model immediately stands out with only two FN. The scratch and pre-trained model on Cats vs. Dogs had seven and six FN, respectively, both at least three times the amount as Inception-ResNet-v2. However, comparing the scratch model with the model pre-trained on Cat vs. Dogs, that difference is almost negligible and may not be statistically significant.

The Inception-ResNet-v2 model is the most reliable and preferred for a consequential classification task like detecting brain tumors.

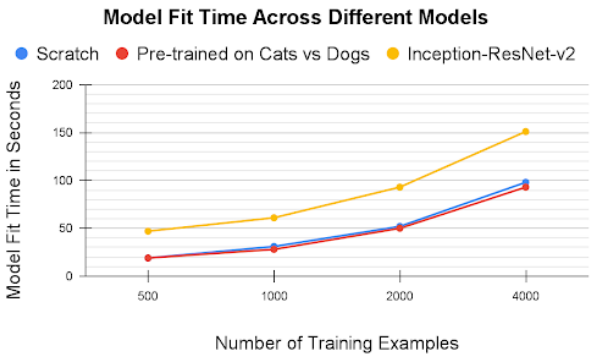

Figure 3.

While we can theoretically allocate ever-increasing amounts of resources like computational power and time to any task, we have to face the fact that there are diminishing marginal returns. As observed in Figure 1, initial increments in training set size improved testing accuracies quickly, but it tapers off just as fast (with a natural upper bound of 100

We compiled the model fit time across different models with different dataset sizes in Figure 3. We have the number of training examples on the horizontal axis, starting at 500 and doubling with each increment. On the vertical axis, we have the model fit times in seconds. The legend shows three lines, each illustrating the change in fit time of a particular model as the number of training examples increases.

We immediately notice that the Inception-ResNet-v2 model takes longer to fit than the other two regardless of the number of training examples–about twice as long at 500 examples. The difference, however, diminishes with more training examples. With 4000 examples, this difference ratio reduces to about 1.5. Additionally, the difference in fit time compared to the other two models has a minor divergence as training examples increase. For the scratch model and the model pre-trained on Cats vs. Dogs, the difference in fit time is almost negligible with any number of training examples.

These results are not surprising. Fit times are largely influenced by two factors: the size of the dataset and the complexity of the model. As mentioned, Inception-ResNet-v2 is far more complex of a model than the other two.

However, we now have to question the scalability of complex pre-trained models. While Inception-ResNet-v2 was undoubtedly the best of the three models in terms of testing accuracy, we also see from Figure 2 that it undoubtedly takes the longest to fit. As we saw, the ratio of the fit time of Inception-ResNet-v2 compared to the other models is approximately 1.5:1.

Referring back to Figure 1, at 4000 training examples, the Inception-ResNet-v2 model was only about a 6

Due to the nature of a task like detecting brain tumors, we are probably willing to dedicate more resources to 1) reduce the number of FN and 2) maximize overall accuracy. Of the three models we tested, Inception-ResNet-v2 accomplishes both of these goals the best.

7. Social Implications

Transfer learning can benefit machine learning classification tasks where datasets do not have abundant examples. When working specifically with the medical field, we, as machine learning engineers, need to be thoughtful about which performance metrics are the most important to optimize, which would involve fine-tuning our models. For example, trying to achieve high training accuracy may not be entirely sufficient, as it may be even more critical to

minimize the number of FN. As such, we need to be very careful about where our focus should be when dealing with potential human lives.

8. Conclusions and Future Directions

Of the three models we tested, Inception-ResNet-v2 boasted the highest testing accuracy across all training set sizes. Not only that, comparing the best versions of each model (when they were all trained with 4000 examples), Inception-ResNet-v2 produced the fewest FN, with the second-best exhibiting three times as much. While Inception-ResNet-v2 comes out on top on these two measures, its performance advantages also come at a cost: it simultaneously takes the longest to fit. Since it took about 1.5 times longer than the other two models, we naturally have scalability concerns:

- Will the ratio of fit times between Inception-ResNet-v2 and the other models maintain as we scale? (1.5:1 is a big difference if we talk about weeks, months, etc.)
- With vastly larger datasets, would the extra cost in computational power and time be worth the marginal benefits?
- How close will the performance of the other two models be to Inception-ResNet-v2 as our training size increases?

We would need to have more than 4000 examples in our training set to answer these questions. We can either use data augmentation on the current brain tumor scans to artificially increase the number of examples or find a different dataset with more examples. Even if we choose to use data augmentation for this particular task, it would still make sense to work with other datasets to see if we get similar results with our three models.

However, assuming these trends continue, it may not be a bad idea to use a simpler pre-trained model or even a model built from scratch if schedules were tight. Although these do fall behind Inception-ResNet-v2 in testing accuracy, the gap closes quickly with increasing training examples.

However, if labeled examples were challenging to come by, using complex pre-trained models is the way to go. Even with longer training times, the Inception-ResNet-v2 model had over 95

For our experiments, we arbitrarily chose several of our hype-parameters. If we trained each model on more than ten epochs, it would have more chances to update its weights. It is feasible to do the same experiments we have done but train each model on twice as many epochs (the exact number is somewhat arbitrary). However, our models may overfit the training data if we choose too many epochs instead. Again, hyper-parameter tuning was not the goal of our experiments, as we cared more about general trends and behaviors. However, if we were to implement any CNN for practical use, we would want it to perform the best it possibly can for that task, so diving deeper into fine-tuning would be worthwhile.

References

Brownlee, J. A gentle introduction to transfer learning for deep learning., 2019. URL https://machinelearningmastery.com/transfer-learning-for-deep-learning/. Last accessed 9 December 2021.

Brownlee, J. Transfer learning in keras with computer vision models., 2020. URL https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-conv Last accessed 9 December 2021.

Brownlee, J. How to classify photos of dogs and cats (with 97 URL https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to- Last accessed 9 December 2021.