Deep Learning Research: Effectiveness of Transfer Learning

Yuan Zhou

Abstract

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. In this research, we built a CNN model trained by Tiny-ImageNet[1], and used the trained model to solve the problem Dogs-vs-Cat[2], and we also used a pre-trained VGG-16 model as a benchmark to evaluate the performance of the best model we generated.

1. Introduction

We use transfer learning to save time or for getting better performance as it is an optimization. In our research, we would like to solve a binary classification problem: Dogs vs Cats, with a dataset of 2,000 images in training and 1,000 images in the test dataset. Such relatively small scale datasets often limit the potential accuracy however the models are built without pre-trained weights, because eventually the overfitting problems occur if we train for too many epochs. In such cases, transfer learning could be used as a rescue. To validate this theory, we first trained the models with Tiny-ImageNet dataset, which is a more

[1] https://tiny-imagenet.herokuapp.com/
[2] https://www.kaggle.com/c/dogs-vs-cats/overview

complex dataset, and then transferred the pre-trained model to the Dogs-vs-Cats problem to achieve a better performance result.

2. Literature Review

Han, Dongmei et al. (2018) created a method and proved that the useful feature presentation of pre-trained network can be efficiently transferred to target task, and the original dataset can be augmented with the most valuable Internet images for classification. Their method not only greatly reduces the requirement of a large training data, but also effectively expands the training dataset. Both of method features contribute to the considerable over-fitting reduction of deep CNNs on small dataset. [1]

Mahbub Hussain et al. (2018) proposed the study and investigation of a CNN architecture model (i.e. Inception-v3) to establish whether it would work best in terms of accuracy and efficiency with new image datasets via Transfer Learning. The retrained model is evaluated, and the results are compared to some state-of-the-art approaches. [2]

3. Methodology

3.1 Research Design

The problem we are about to solve is to build a model that could classify dogs vs. cats with satisfying accuracy. The data we have at hands are:

- dogs vs. cats dataset
- Additional training dataset: Tiny ImageNet.

The details of these two datasets are covered in section 3.2.

To explore and build the best model making most use from our two datasets, we have designed the research to contain four phases:

- Phase 1: build and train 1 - 5 blocks of Convnet+MaxPooling layers, with the dogs-vs-cat data directly.
- Phase 2: build and train another 5 models with the same architectures in Phase 1, and train with the Tiny ImageNet dataset; persist the trained models to a file to complete transfer learning in Phase 3.
- Phase 3: Create 5 models using CNN layers persisted in Phase 2 as transferred layers and finish the training with the dogs-vs-cats dataset; compare the predictions by these models versus the ones generated by models in Phase 1.
- Phase 4: compare the results with benchmark VGG16; optimize the model generated in Phase 3, and make some visualizations to understand the model's behavior.

3.2 Data Understanding and Preparation

We used two datasets in this research, and the following are some key facts about them.

Dogs vs. Cat: this dataset has 2000 training images, with 1000 of class Dog, and 1000 of class Cat; and the test set contains 1000 images. Each image is of 256*256 pixels resolution.

Tiny ImageNet: 100,000 images in the training set, spread evenly across 200 classes (each class has 500 images); test set contains 10,000 images. Each image is of 64*64 pixels resolution.

Because of the good amount of data volume, and great balance between classes for each dataset, we did not pre-process much of the original dataset. To fight potential overfitting issue, we used this specific image generator configurations:

train_datagen=ImageDataGenerator(rescale=1.0/255.0,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True)

3.3 Modeling Method

The general principle we follow when building the models was to add layers/Convnet blocks carefully with solid justification based on training and test results. From CNN-1 to CNN-5, we increased one block of 1xConv2D + 1xMaxPooling gradually, and tested their performance accordingly. We kept the same dense layers (Flatten + Dense(128) + Output(2)) and other hyperparameters to make sure the difference was from architectural variance.

In CNN-6 and CNN-7, we focused on increasing more Con2D layers within a block. So, in CNN-7, it has 4 blocks of (2xConv2D + 1xMaxPooling) and we trained and tested with the same hyperparameters and dataset.

For all optimizations, the general principle was still followed and we collected solid performance data when making every new modification in our best model's architecture.

3.4 Visualization

For each model explored in the experiment, we used at least two diagrams to visualize their performance from two aspects:

- A plot diagram to visualize the correlation between training/validation accuracy and epochs
- A plot diagram to visualize the correlation between training/validation loss and epochs

Plus, we also drew feature maps and heatmaps for our best model and VGG16 as a comparison analysis.

4. Results

4.1 CNN models training directly with dogs-vs-cats dataset

In research step 1, we created five different models, varying based on their depth; the detailed architecture comparison between

them are in Appx-Table-1. In the training of these models, we kept some hyperparameters the same to draw a fair comparison specifically related to their differences in architecture:

- Training epochs: 10
- Conv2D layers: activation='relu', kernel_initializer='he_uniform', padding='same'
- Output layers: activation='sigmoid'
- Optimizer: SGD(lr=0.001, momentum=0.9)

Model	No. of Conv. blocks	No. of Conv Layers	Training time per epoch (seconds)	Test Accuracy (%)	Test Loss
CNN-1	1	1	29	50.00	0.6931
CNN-2	2	2	45	53.60	0.6861
CNN-3	3	3	60	60.70	0.6555
CNN-4	4	4	73	62.20	0.6713
CNN-5	5	5	80	56.10	0.6885
CNN-6	4	6	188	62.90	0.6676
CNN-7	4	8	242	69.30	0.5856

Table-1 Perf. metrics of CNN-1,2,3,4,5,6,7

The key facts revealed by Table-1 are:

- Training time steadily increases when the model's CNN layers count increases; i.e. deeper learning models are taking more time to train.
- The test accuracy improved when Convnet blocks (x number of Conv2D layers plus one MaxPooling layer) increased from 1 to 4; accuracy of 5 Convnet blocks decreased when comparing to 4 Convnet blocks; thus we decided to use 4 Convnet blocks as the most suitable number.
- CNN-6 and CNN-7 also have 4 Convnet blocks, but their Conv2D layers per block increased when compared to CNN-4; and the training result shows the accuracy improves when each Convnet block has more Conv2D layers.
- The best accuracy result achieved by these CNNs is 69.30%, which could be used as a baseline to compare with the accuracy achieved by transfer learning models in 4.2 and 4.3.

4.2 Compare performance of transfer learning Models

In Step 2, we created similar models to what we had in Step 1, by editing their dense layers to fit the 200 classes classification for Tiny-ImageNet dataset. We trained the models in Step 2 with Tiny-ImageNet dataset for 25 epochs, and saved their models architecture and weights as the CNN layers for transfer learning. As a comparison to benchmark, we also leveraged a VGG16 model pre-trained by ImageNet dataset to create a transferred model. The final performance results generated for those models are generated below:

Model	No. of Conv Layers	Training time per epoch (seconds)	Original model Test Accuracy (Tiny-Image Net) (%)	Transferred Model Test Accuracy (dogs-vs-cats) (%)
CNN-1	1	65	20.86	76.00
CNN-2	2	72	27.84	77.80
CNN-3	3	85	32.65	79.40
CNN-4	4	102	34.97	80.90
CNN-5	5	136	35.37	79.40
VGG16	16	N/A	N/A	97.400

Table-2 Perf. metrics of CNN-1,2,3,4,5 and VGG16

Table-2 revealed some key findings:

- The training time comparison: CNN-1 < CNN-2 < CNN-3 < CNN-4 < CNN-5. This concurred the observation we had in 4.1: more complex and deeper models take longer time to train.
- Test accuracy when training Tiny-ImageNet: CNN-1 < CNN-2 < CNN-3 < CNN-4 < CNN-5. This result is similar compared to what we had in 4.1 when training these models with dogs-vs-cats, but when training with Tiny-ImageNet, CNN-5 behaves better than CNN-4. This is likely because that Tiny-ImageNet has 50x more data, and 100x more classes than dogs-vs-cats, which means it's much more complex -- and CNN-5 behaves better than CNN-4 in such a dataset because of its complexity.
- Transferred Model Test Accuracy using dogs-vs-cats: after the training with Tiny-ImageNet dataset, even the simplest model, CNN-1, performs much better (76%)

than the CNN-7 when training with dogs-vs-cats (69%). This is strong evidence that the quality of training dataset is the key to a model's performance. Due to its advantage over dogs-vs-cats, the models pre-trained with Tiny-ImageNet are capable of handling a problem of smaller scale in terms of dataset size & complexity.

- The comparison of transferred models test accuracy also shows that there is a sweet-spot when adding layers to the model architecture: CNN-4 behaves the best and beats CNN-5 with around 1.5%, while its model has fewer Convnet layers than CNN-5. This is also an indicator that we could make some optimizations attempts based on the architecture of CNN-4 and build our best model.
- VGG16 as a benchmark provides 97.40% accuracy, which beats all our models easily. We believe the biggest difference is the data quality: we transferred the weights pre-trained based on the original ImageNet dataset, which has over 14 million images and 1,000 classes. As a much more complex model, VGG has 16 layers of Convnet and of 5 Convnet blocks, which is suited to leverage such a large dataset.

5. Optimizing CNN model performance

According to the results analyzed in Section 4, we decided to use CNN-4 model as the base to optimize. We applied a few attempts with following approaches:

- Training with more epochs and steps: we increased the total epochs and steps per epoch to make sure there’s no underfitting problem.
- Apply dropout layers to fight overfitting problems.
- Add additional Convnet layers within each Convnet block to find out the most suitable depth of the model.

The adjusted model's performance are listed here:

Model	Optimizations	Original model Test Accuracy (Tiny-ImageNet) (%)	Transferred Model Test Accuracy (dogs-vs-cats) (%)
CNN-4	None	34.97	80.90
CNN-4-1	2x epochs 2x steps Dropout (0.2)	34.63	81.50
CNN-4-2	2x epochs 2x steps 2xConvnets per block	36.68	81.30
CNN-4-3	2x epochs 2x steps Dropout (0.2) 2xConvnets per block	36.47	79.50

Table-3 Perf. metrics collected for optimized models

From Table-3 we can tell:

- Increasing the training epochs and steps per epoch improved the final test accuracy: the dataset was utilized because of the training.
- CNN-4-2 and CNN-4-3 are more complex and generated better performance in Tiny-ImageNet dataset; but CNN-4-1 is still the best model in terms of prediction accuracy for dogs-vs-cat dataset; so we chose the CNN-4-1 as our best model after optimization.

6. Visualize the best model

The best model we finally built was illustrated in Appx-Fig-1. The final model has 4 Convnet (1xConv2D + 1xMaxPooling layers) blocks, followed by a Dropout layer after each block. After the CNN blocks, we finally have a Flatten layer and two Dense layers to get the classification output.

In Appx-Fig-2, we have illustrated the feature map of CNN-4-1. We can see that the laste CNN layer with 512 filters has very sparse features, when compared to VGG-16's features map. This is because the convolution layers are followed by a ReLU activation layer that turns all negative inputs into zeros, making the output (i.e., feature maps) of the CNN layer highly sparse.

The heatmap comparison between CNN-4-1 and VGG-16 for some test images samples shows that the feature activations are more understandable in VGG-16 to humans. This

concurs with the fact that VGG-16 has better performance than CNN-4-1 for dogs-vs-cats classification.

7. Conclusion

From the previous experiments we did and analysis of the models, we can draw a conclusion that the CNN models pre-trained with a large scale of dataset could result in a better performance than training the model directly with a small scale of dataset.

References

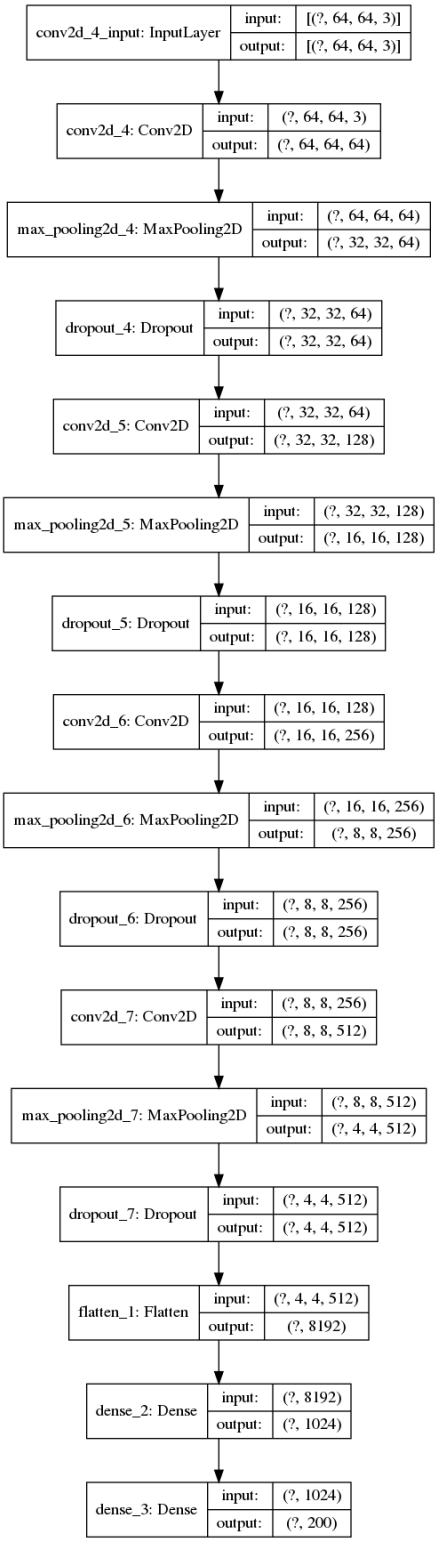
[1] Han, D., Liu, Q., & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. Expert Systems with Applications, 95, 43-56.

[2] Hussain, M., Bird, J. J., & Faria, D. R. (2018, September). A study on cnn transfer learning for image classification. In UK Workshop on Computational Intelligence (pp. 191-202). Springer, Cham.

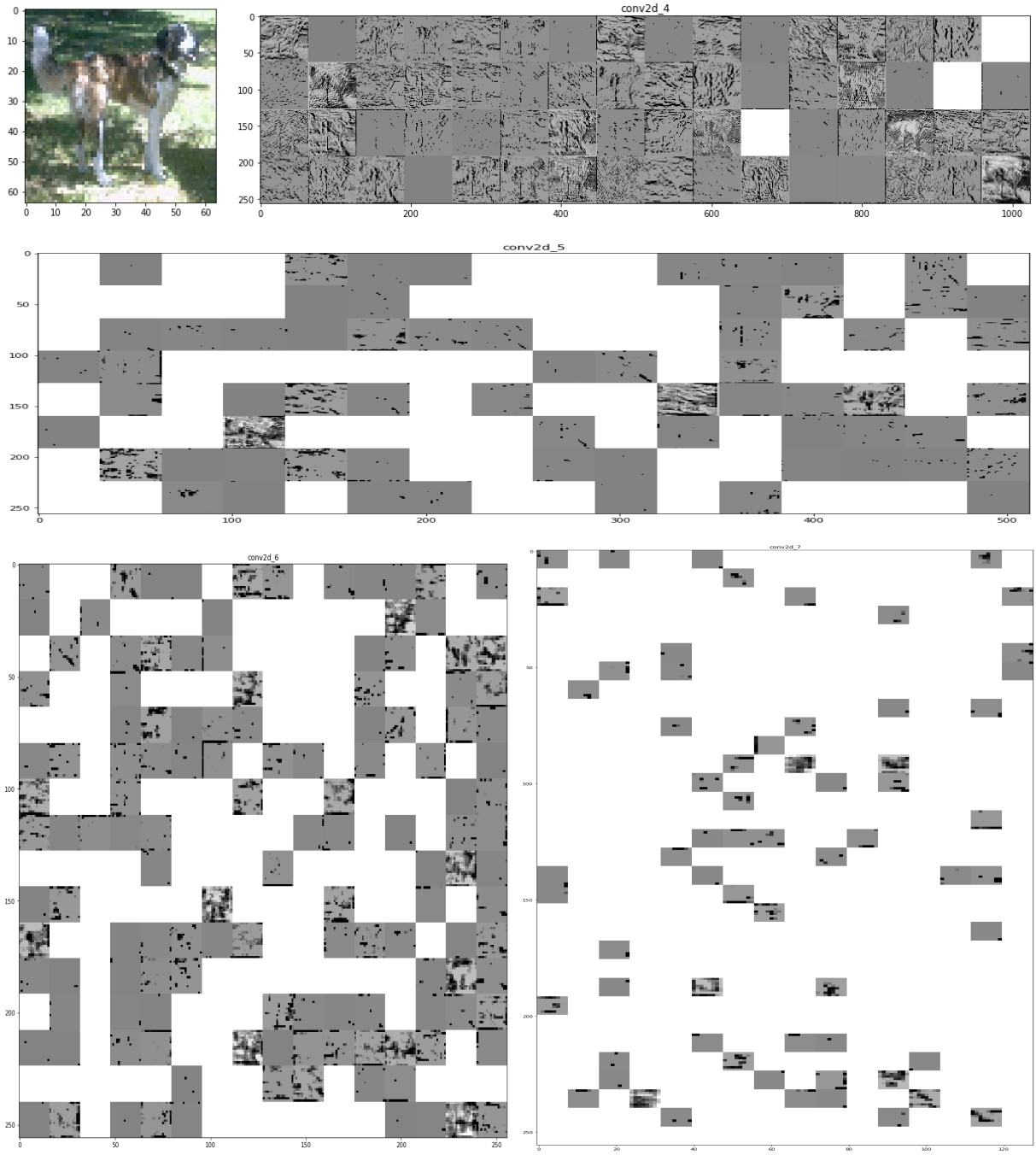
Appendix

Model Name	CNN-1	CNN-2	CNN-3	CNN-4	CNN-5	CNN-6	CNN-7
Input Layer	200 x 200 RGB Image						
Convolution Layers	1xConv2D-64 MaxPooling	1xConv2D-64 MaxPooling 1xConv2D-128 MaxPooling	1xConv2D-64 MaxPooling 1xConv2D-128 MaxPooling 1xConv2D-256 MaxPooling	1xConv2D-64 MaxPooling 1xConv2D-128 MaxPooling 1xConv2D-256 MaxPooling 1xConv2D-512 MaxPooling	1xConv2D-64 MaxPooling 1xConv2D-128 MaxPooling 1xConv2D-256 MaxPooling 1xConv2D-512 MaxPooling 1xConv2D-512 MaxPooling	2xConv2D-64 MaxPooling 2xConv2D-128 MaxPooling 1xConv2D-256 MaxPooling 1xConv2D-512 MaxPooling	2xConv2D-64 MaxPooling 2xConv2D-128 MaxPooling 2xConv2D-256 MaxPooling 2xConv2D-512 MaxPooling
Flatten Layer	Flatten						
Dense Layers	1xDense-128						
Output Layer	1xDense-1						

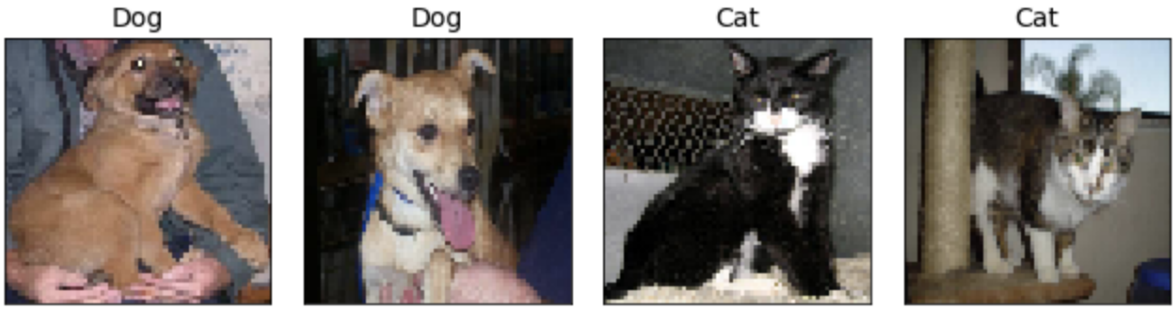
Appx-Table-1 CNN models created for dogs-vs-cats data training



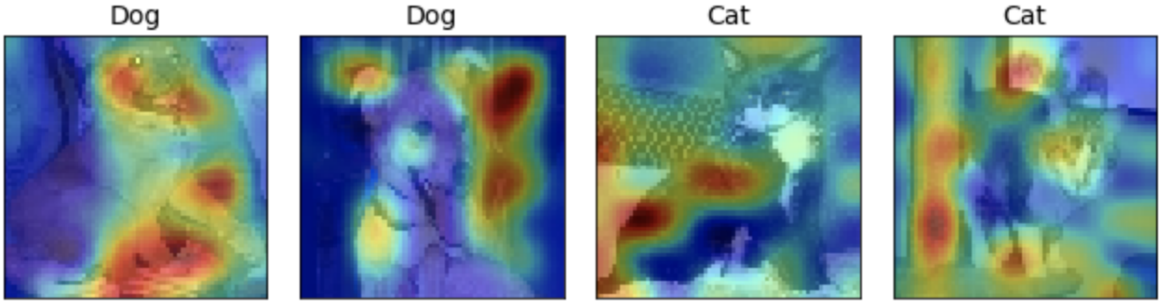
Appx-Fig-1 CNN-4-1 model architecture diagram



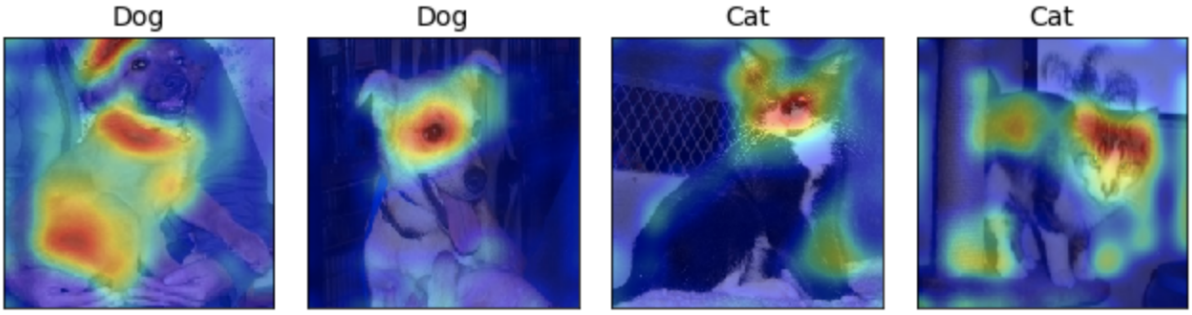
Appx-Fig-2 Feature Maps of CNN-4-1 model's prediction



Original Images



CNN-4-1 Heatmap



VGG-16 Heatmap

Appx-Fig-3 Heatmap comparison between CNN-4-1 and VGG-16