CS577 Assignment 4: Cats vs Dogs Report

Theo Guidroz
Department of Computer Science
Illinois Institute of Technology

April 29, 2021

Abstract

This is a report for the first programming question of Assignment 4. It serves as an introduction to transfer learning and the use of data generators to feed the model.

Problem Statement

Build two data generators; one with and one without data augmentation. Train a first simple CNN model on the dataset with no data augmentation, a second model with a frozen VGG16 and a third model with the top of VGG16 unfrozen on the same dataset. Lastly, train a frozen VGG16 model on augmented data.

Implementation details

This question was divided into 7 tasks:

1. Loading the dataset
2. Building the data generators
3. Building the simple CNN model
4. Building the frozen VGG16 model
5. Building the VGG16 model without freezing the top layer
6. Training the models
7. Evaluating them

Loading the datasets

It is important to ensure that the data provided to the model is of the highest quality. To do so, the 2000 heaviest images of each label were loaded in the datasets. First, the other pictures were manually deleted and then, the 4000 images were renamed 0 to 1999. Each folder now contains 2000 images. The idea of using only the heaviest images came up when I encountered an issue when trying to load a file which was 0KB. Now that we have a dataset of high quality, we can go to the next section.

Building the data generators

Data generators make it easier to feed images to the model. Using the example shown in class, we create the generators with no augmented data using:

```
def GetDatagen():
    test_datagen = ImageDataGenerator(rescale=1./255)
    train_datagen = ImageDataGenerator(rescale=1./255)
    return train_datagen,test_datagen
def Generator_without_Augmentation():
    train_datagen,val_datagen = GetDatagen()
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(150,150),
        batch_size=32,
        class_mode='binary')

    validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(150,150),
        batch_size=32,
        class_mode='binary')
    return train_generator,validation_generator
```

We can now feed the model using fit_generator. The generator for augmented data is similar but we modify the train_datagen to:

```
train_datagen = ImageDataGenerator(
            rescale=1./255,
            rotation_range=40,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')
```

Now, we have both of our datasets built.

Building the CNN model

For this classification problem, we build a simple CNN model with 4 convolution and max pooling block followed by batch normalization to help the model generalize. We then flatten the data and perform drop out. The last activation used is sigmoid since it is a binary classification problem.

Building the frozen VGG16 model

This model is built using VGG16 available on Keras and using imagenet as weight. We then flatten the data, perform dropout and add a layer with sigmoid activation. We then set layers.trainable to false to freeze the weights.

Building the VGG16 model without freezing the top layer

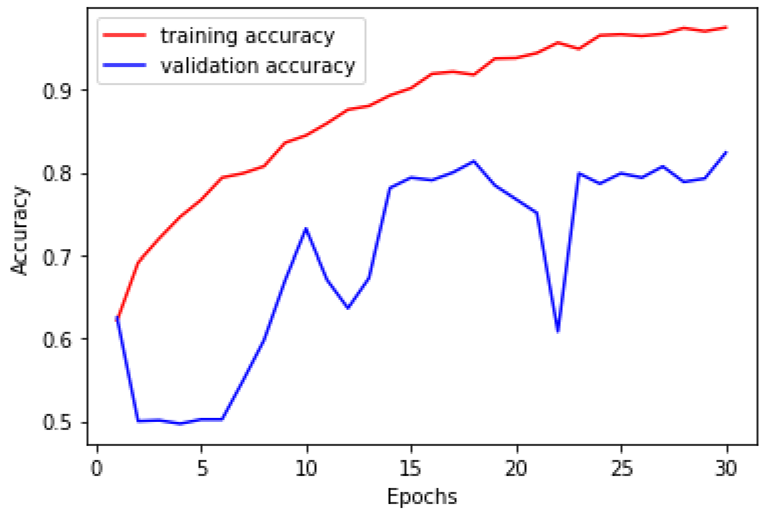
This model is built off the previous one but we unfreeze the top layers. This is done using the following code:

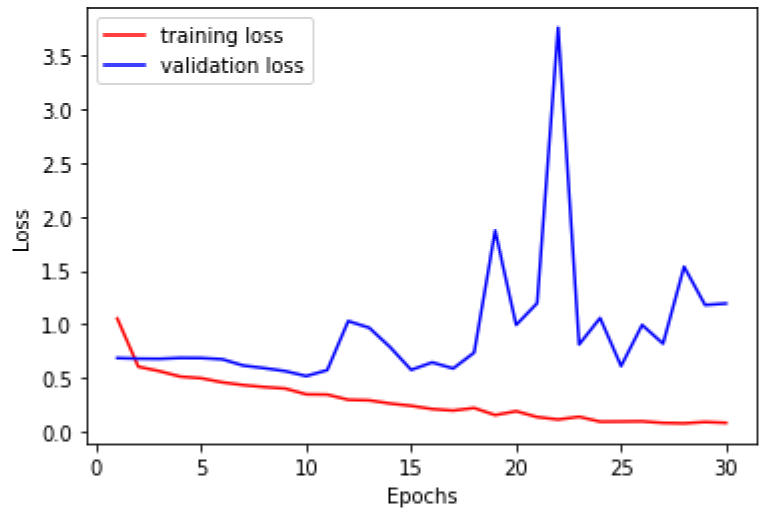
```
for layer in conv_base.layers:
    if layer.name == "block5_conv1":
      set_trainable = True
    if set_trainable:
      layer.trainable = True
    else:
      layer.trainable = False
```

Now we can train this model and the top weights of VGG16 will be modified.

Results and discussion

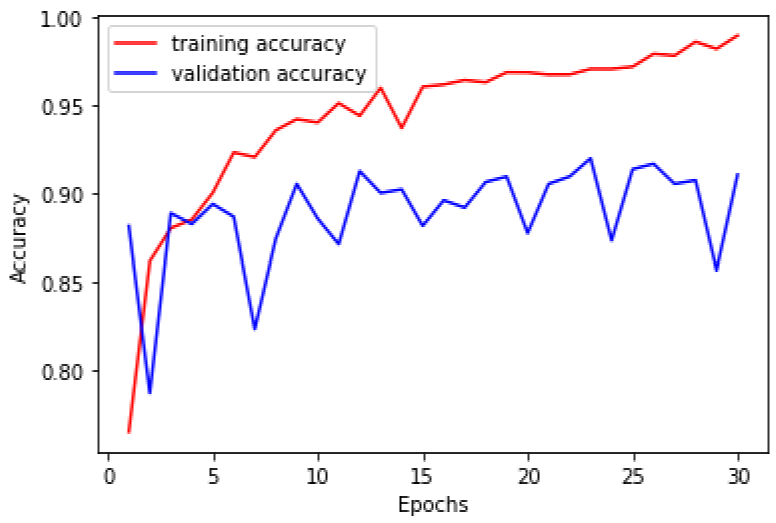
Model 1: Simple CNN

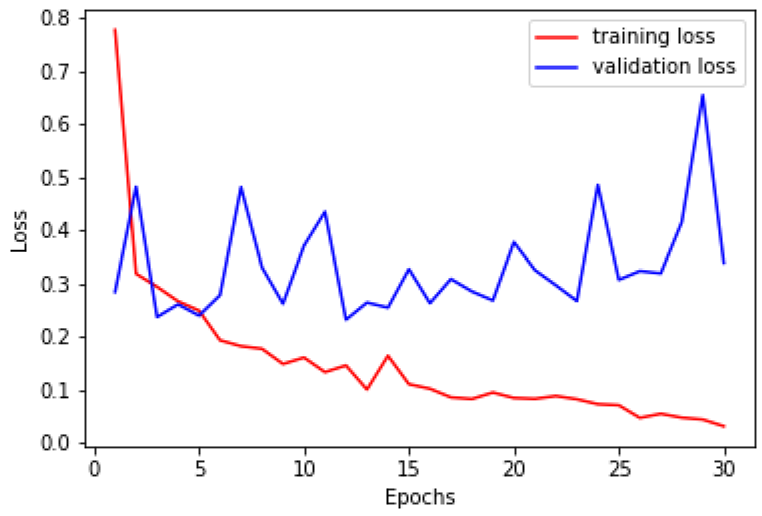




The graphs above show the loss and accuracy of the model against the number of epochs. The model converges but there are signs of overfitting since training loss is decreasing but validation loss is increasing.

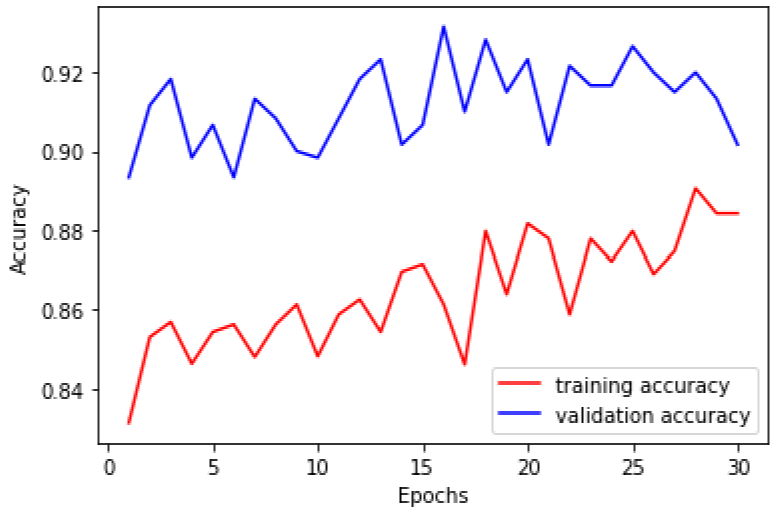
Model 2: Frozen VGG16 on normal dataset

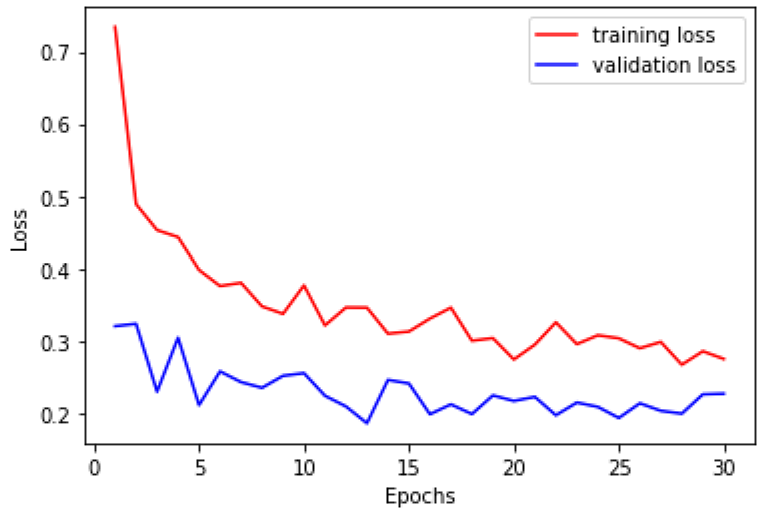




While the training loss decreases, the validation loss does not. This is a sign that the model is overfitting.

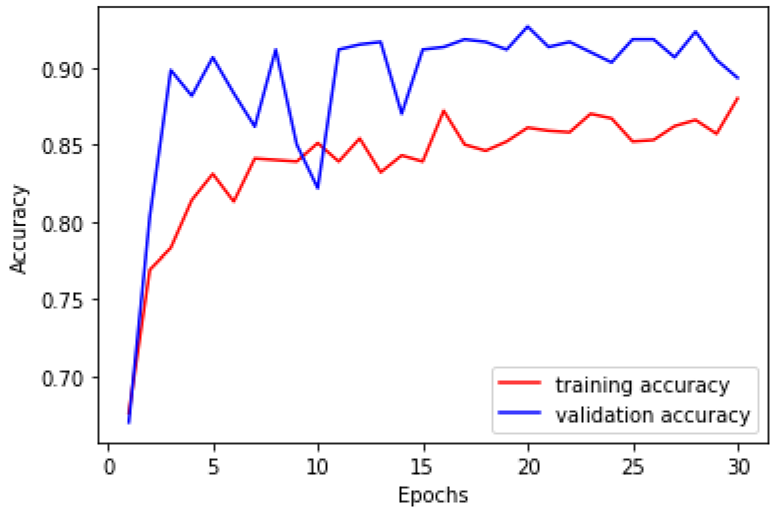
Model 3: Training the same model but unfreezing some layers

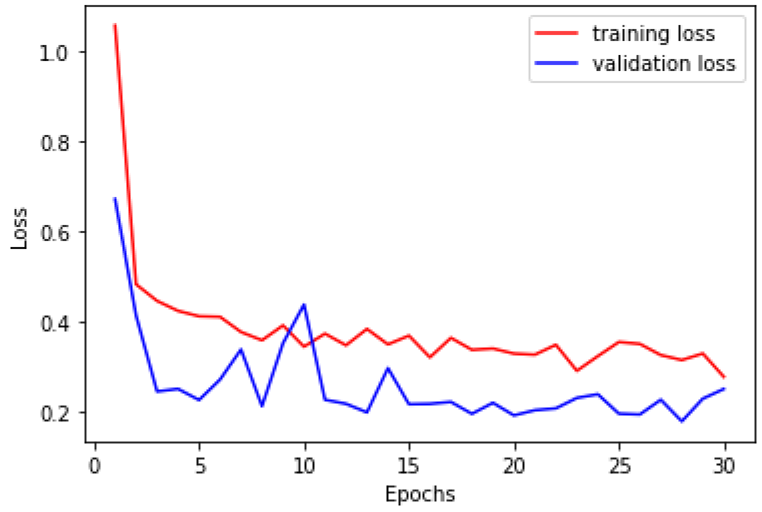




The graph shows that the model is converging, since both losses are decreasing and the accuracies are increasing.

Model 4: Frozen VGG16 on augmented dataset





The graph shows that the model is converging, since both losses are decreasing and the accuracies are increasing.

Comparing the models

	Model1	Model2	Model3	Model4
0	0.625000	0.881250	0.893333	0.670000
5	0.502083	0.886458	0.893333	0.883333
10	0.670833	0.870833	0.908333	0.911667
15	0.790625	0.895833	0.931667	0.913333
20	0.751042	0.905208	0.901667	0.913333
25	0.793750	0.916667	0.920000	0.918333
29	0.823958	0.910417	0.901667	0.893333

The number in the left column is the number of epochs. From the table above, we can see that the models which use transfer learning performs significantly better than the simple CNN model.

```
Test acc for model1: 0.5699999928474426%
Test acc for model2: 0.878000020980835%
Test acc for model3: 0.878000020980835%
Test acc for model4: 0.8420000076293945%
```

This is the accuracy of each model when evaluated on the test data. This proves our hypothesis of transfer learning significantly increasing accuracy on limited dataset.

Conclusion

Overall, this assignment was a great example of the benefits of transfer learning. It also shows that augmented data can lead to more generalized models but however, it does not ensure better accuracy.

References

Professor’s slides