Image Classifier: Dog Breed Classification

In-Depth Analysis

The method used to classify the dog breed images is a convolutional neural network (CNN). This was done using Keras and Tensorflow in Python. The simplest model to use in Keras for a CNN is Sequential(), therefore this was used to build the model. An AWS machine was used to increase computational capacity. This allowed for the use of five layers in the CNN. The activation functions used were rectified linear unit (relu) on the first three layers and softmax used for the last layer. This combination provided the best results, figure 1.

The learning rate was set at .1 as this was the smallest learning rate possible with the computational power available. Testing the different hyperparameters also resulted in Adam being the best optimizer.

```
model = Sequential()
model.add(Conv2D(64, (3,3), input_shape = feature_norm.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(120))
model.add(Activation('softmax'))

adam = optimizers.Adam(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(feature_norm, label, batch_size=32, epochs=3, validation_split=0.2)
```

Fig. 1. Best Performing CNN Model

To measure the accuracy of the model, the validation accuracy is used. The training set was split into 80% training data and 20% validation. The accuracy used to assess the model is the validation accuracy.

```
Epoch 1/3
256/256 [==============================] - 375s 1s/step - loss: 284.7138 - accuracy: 0.0097 - val_loss: 280.5924 - val_accur
acy: 0.0073
Epoch 2/3
256/256 [==============================] - 371s 1s/step - loss: 284.7130 - accuracy: 0.0075 - val_loss: 280.5924 - val_accur
acy: 0.0117
Epoch 3/3
256/256 [==============================] - 371s 1s/step - loss: 284.7131 - accuracy: 0.0086 - val_loss: 280.5924 - val_accur
acy: 0.0103
```

Fig. 2. Results of The CNN Model

The best validation accuracy of the three epochs was 1.17%. For comparison purposes, figures 3 and four show the validation accuracies of the weaker models. CNN Model #2's best result was .88%. This model had all activation functions set to relu. CNN Model #3's best result was .98%. This model used softmax as the last activation layer but only had four layers.

In [36]:
```
model = Sequential()
model.add(Conv2D(64, (3,3), input_shape = feature_norm.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(120))
model.add(Activation('relu'))

adam = optimizers.Adam(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(feature_norm, label, batch_size=32, epochs=3, validation_split=0.2)
```

```
Epoch 1/3
256/256 [==============================] - 367s 1s/step - loss: -824.7269 - accuracy: 0.0082 - val_loss: -835.4014 - val_acc
uracy: 0.0088
Epoch 2/3
256/256 [==============================] - 367s 1s/step - loss: -847.8868 - accuracy: 0.0078 - val_loss: -835.4014 - val_acc
uracy: 0.0088
Epoch 3/3
256/256 [==============================] - 367s 1s/step - loss: -847.8868 - accuracy: 0.0078 - val_loss: -835.4014 - val_acc
uracy: 0.0088
```

Out[36]: <tensorflow.python.keras.callbacks.History at 0x7f138c357b00>

Fig. 3. CNN Model #2

In [35]:

```
model = Sequential()
model.add(Conv2D(64, (3,3), input_shape = feature_norm.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(120))
model.add(Activation('softmax'))

adam = optimizers.Adam(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(feature_norm, label, batch_size=32, epochs=3, validation_split=0.2)
```

```
Epoch 1/3
256/256 [==============================] - 366s 1s/step - loss: 284.7254 - accuracy: 0.0084 - val_loss: 280.5924 - val_accur
acy: 0.0083
Epoch 2/3
256/256 [==============================] - 363s 1s/step - loss: 284.7130 - accuracy: 0.0083 - val_loss: 280.5924 - val_accur
acy: 0.0073
Epoch 3/3
256/256 [==============================] - 364s 1s/step - loss: 284.7130 - accuracy: 0.0067 - val_loss: 280.5924 - val_accur
acy: 0.0098
```

Out[35]: <tensorflow.python.keras.callbacks.History at 0x7f1394281da0>

Fig. 4. CNN Model #3