Predicting Dog Breeds

Third place submission

Strategy

- Initially tried training a CNN from scratch - you can imagine how that went
- Started looking into using an existing pre-trained model
- Came across a tutorial on Kaggle that described Transfer Learning for Images along with this useful guide from Keras

How does transfer learning work?

Learned features from pre-trained neural network

Transfer to a new and different dataset to initialize training

Learning (backpropagation) only happens in the last layers initialized with random weights

Using Inception V3

- Inception V3 is a model that was developed at Google and pre-trained on ImageNet (1.4M images and 1000 classes)

Process:

1) Instantiate a base model and load pre-trained weights into it
2) Run new dataset through it and record the output of layers from the base model (feature extraction)

```
from keras.applications.inception_v3 import InceptionV3, preprocess_input
input_layer = Input(img_size)
preprocessor = Lambda(preprocess_input)(input_layer)
base_model = InceptionV3(weights = 'imagenet', include_top = False, input_shape = img_size)(preprocessor)
avg = GlobalAveragePooling2D()(base_model)
features_model = Model(inputs = input_layer, outputs = avg)
train_features = features_model.predict(X, batch_size = 64, verbose=1)
```

3) Use that output as input data for a new, smaller model

```
finetune_model = keras.models.Sequential([InputLayer(train_features.shape[1:]),
                                          Dropout(0.7),
                                          Dense(n_breeds, activation='softmax')])
finetune_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
finetune_model.fit(train_features, y, batch_size = 128, epochs = 60, validation_split = 0.1, callbacks = my_callback)
```