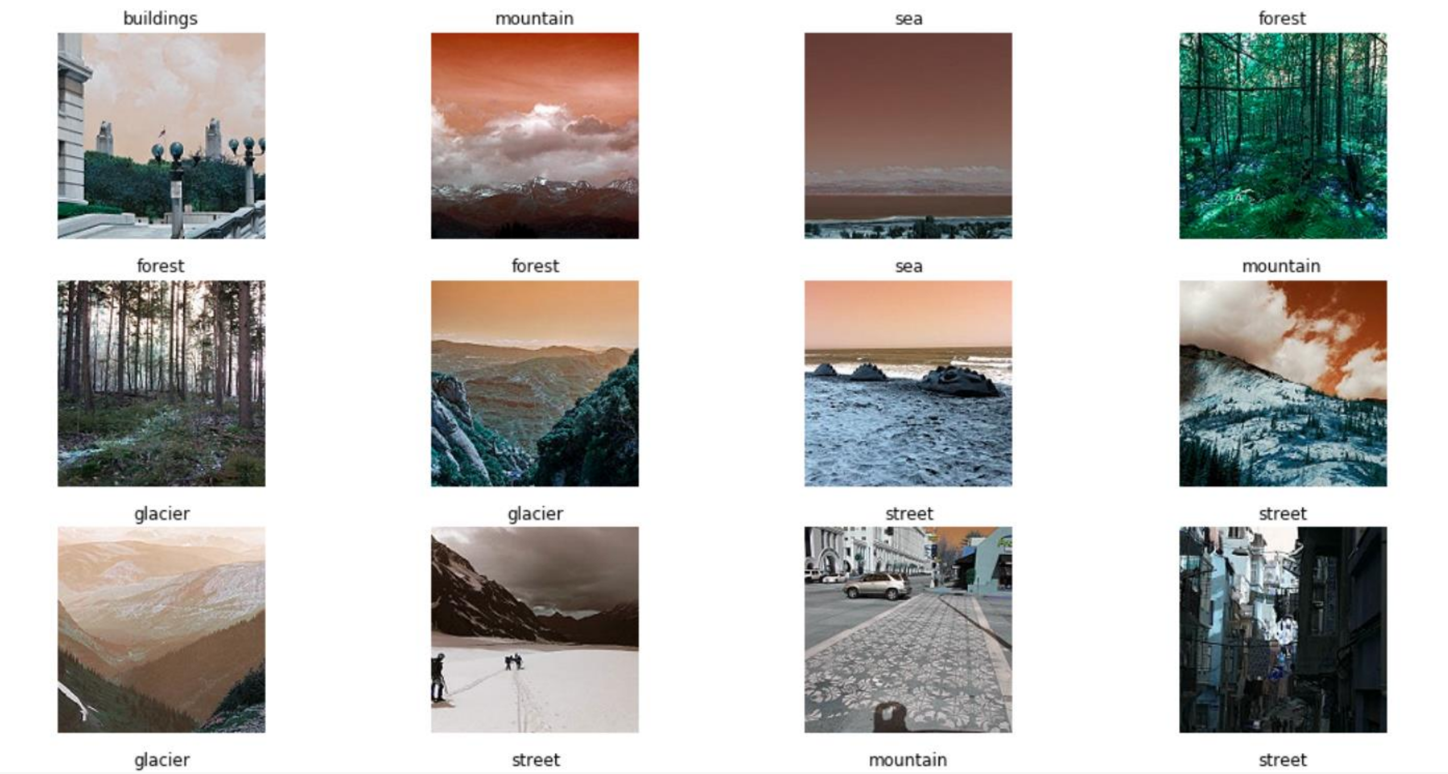
IBM ADVANCED DATA SCIENCE CAPSTONE PROJECT

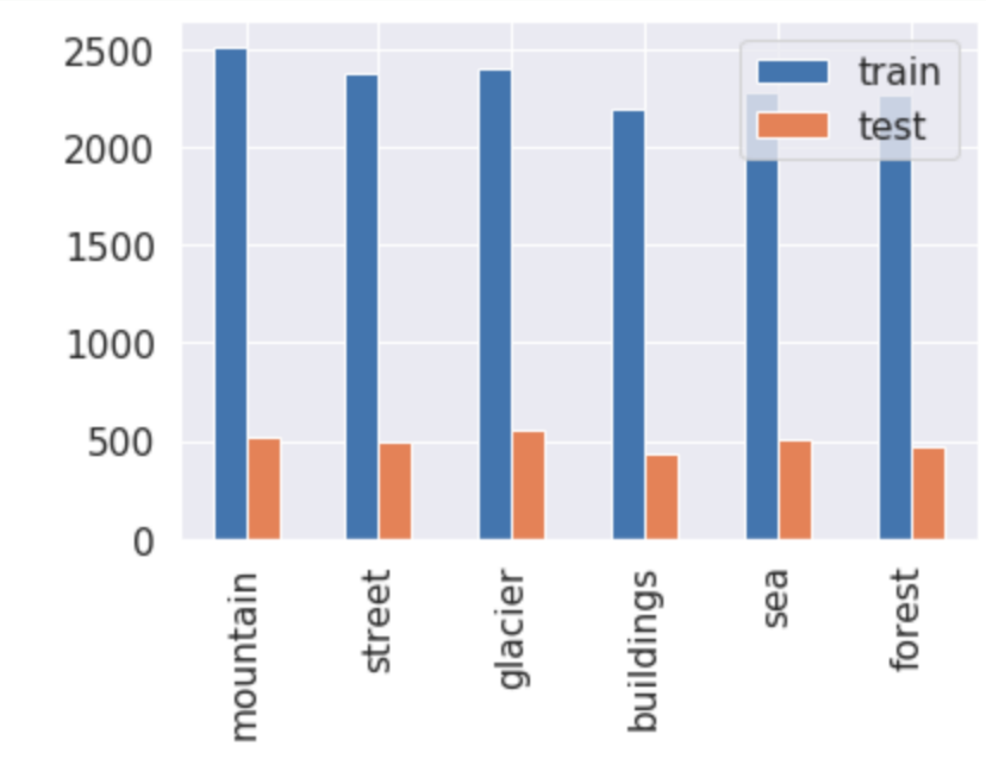
Yaroslav Aulin

Dataset

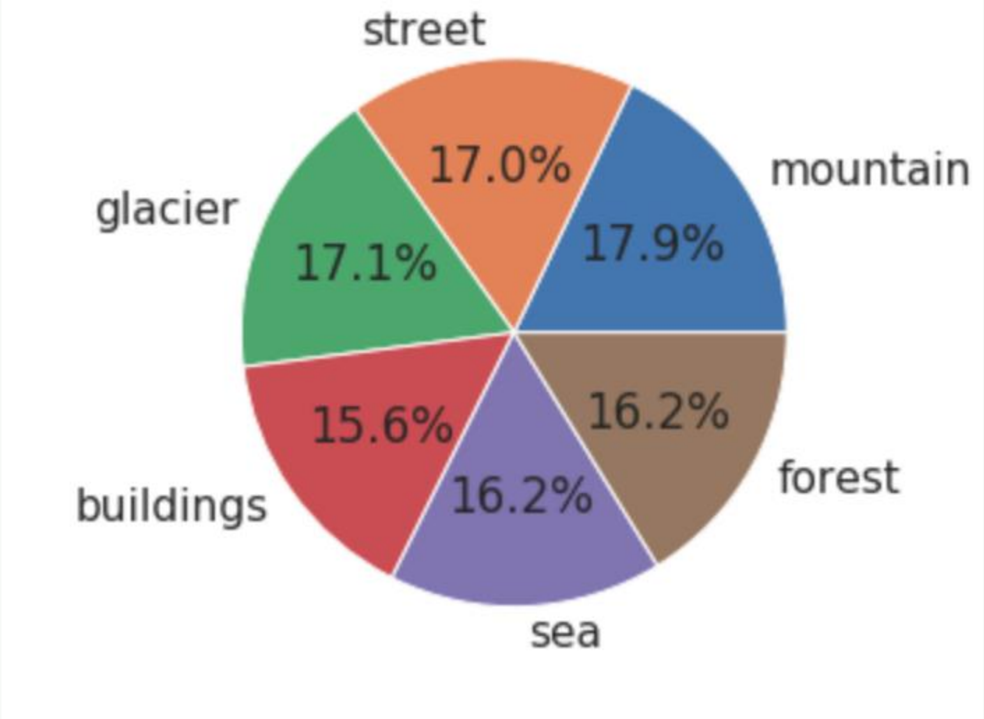
https://www.kaggle.com/puneet6060/intel-image-classification



Exploring Dataset



Train Set

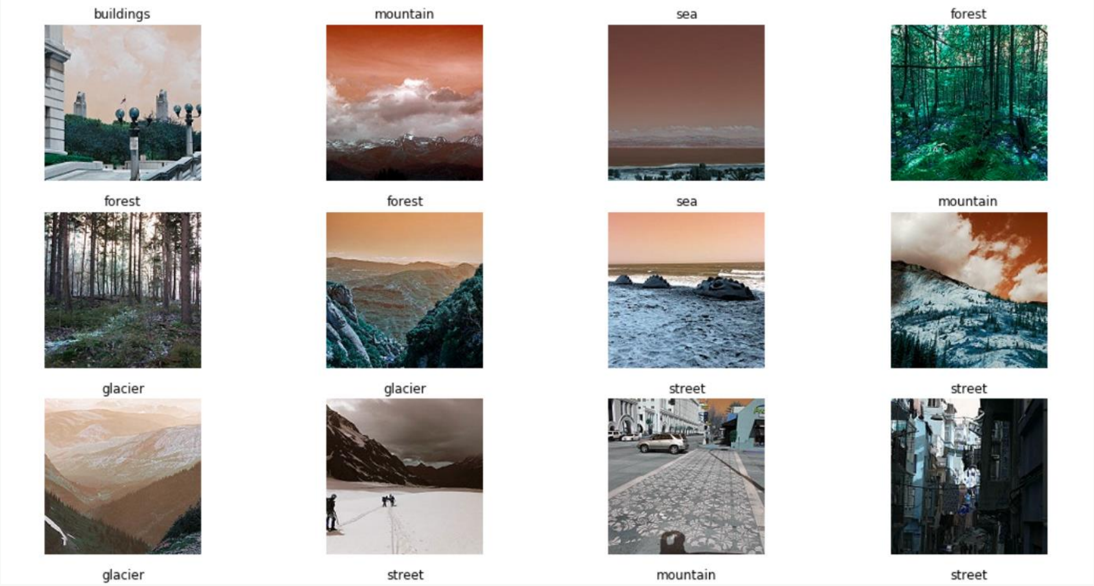


Data Preprocessing

Make all images the same size:

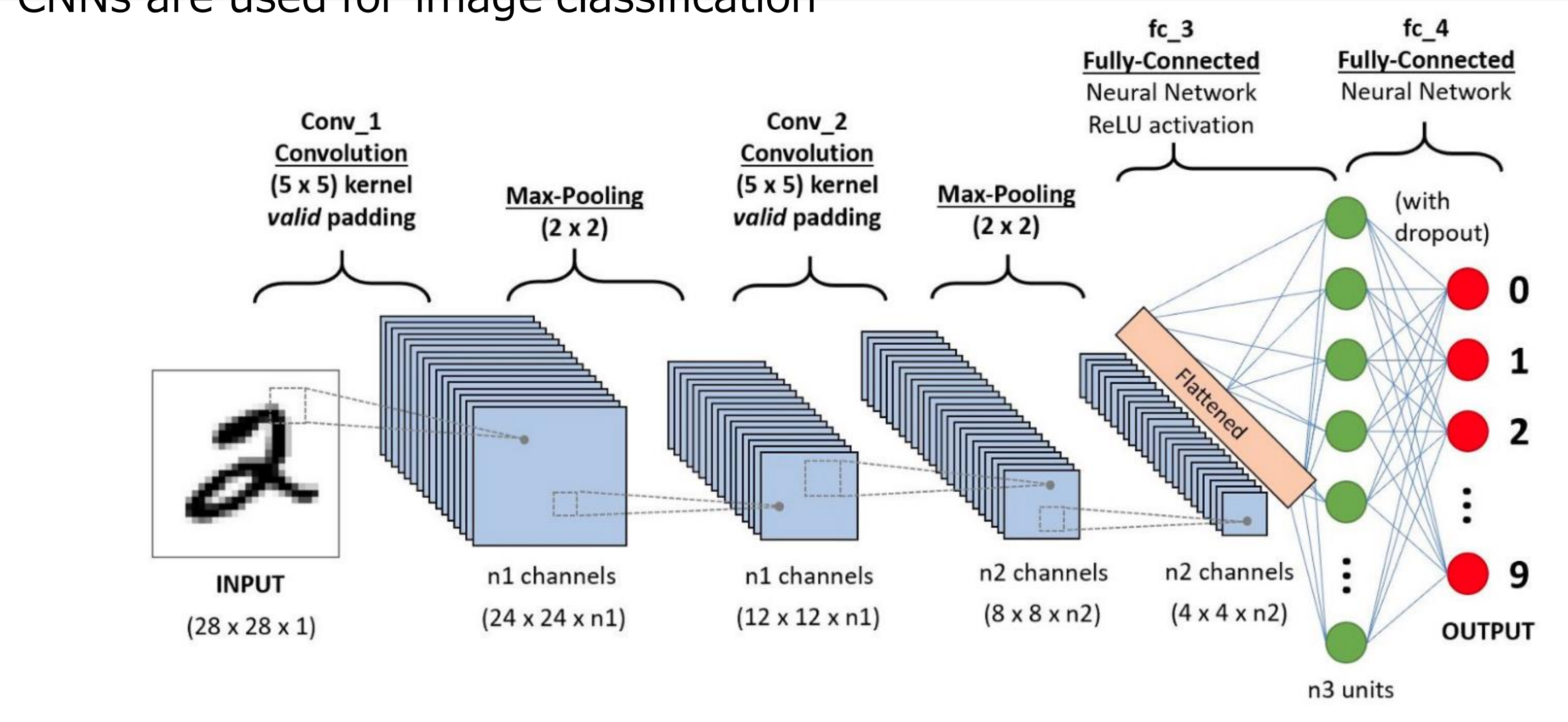
IMAGE_SIZE = (150, 150)

image = cv2.resize(image, IMAGE_SIZE)



Convolutional neural network

CNNs are used for image classification



towardsdatascience.com

Constructing a CNN

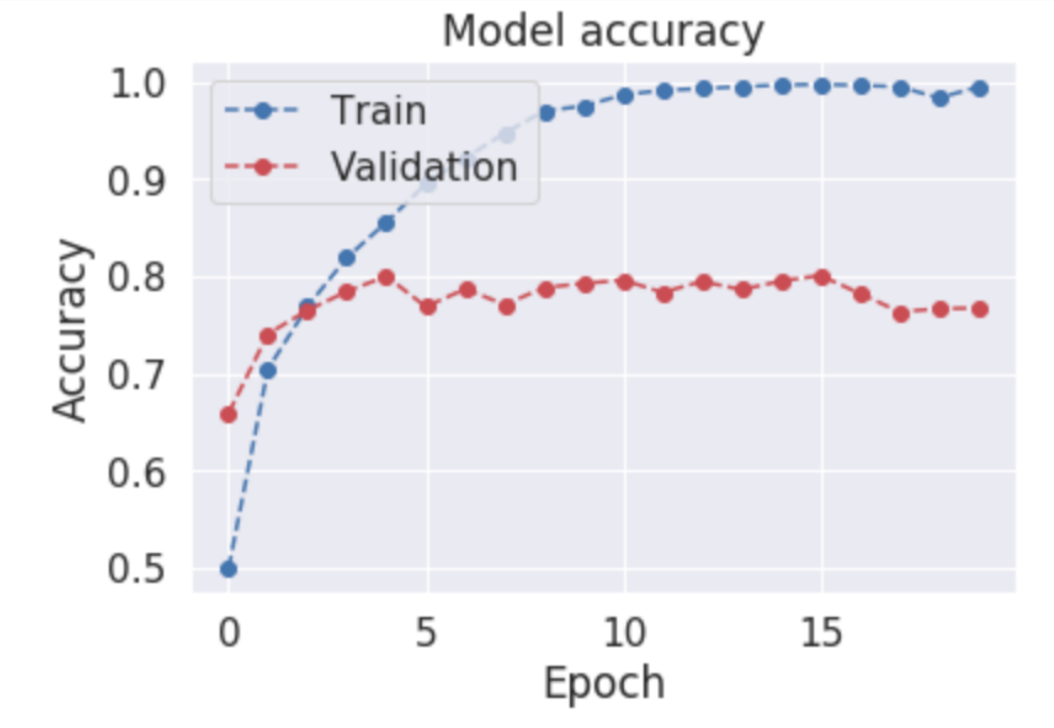
```
# create CNN to predict labels

model = Models.Sequential()

model.add(Layers.Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(150,150,3)))
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.Conv2D(32,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.Flatten())
model.add(Layers.Dense(128,activation='relu'))
model.add(Layers.Dense(6,activation='softmax'))

model.summary()
```

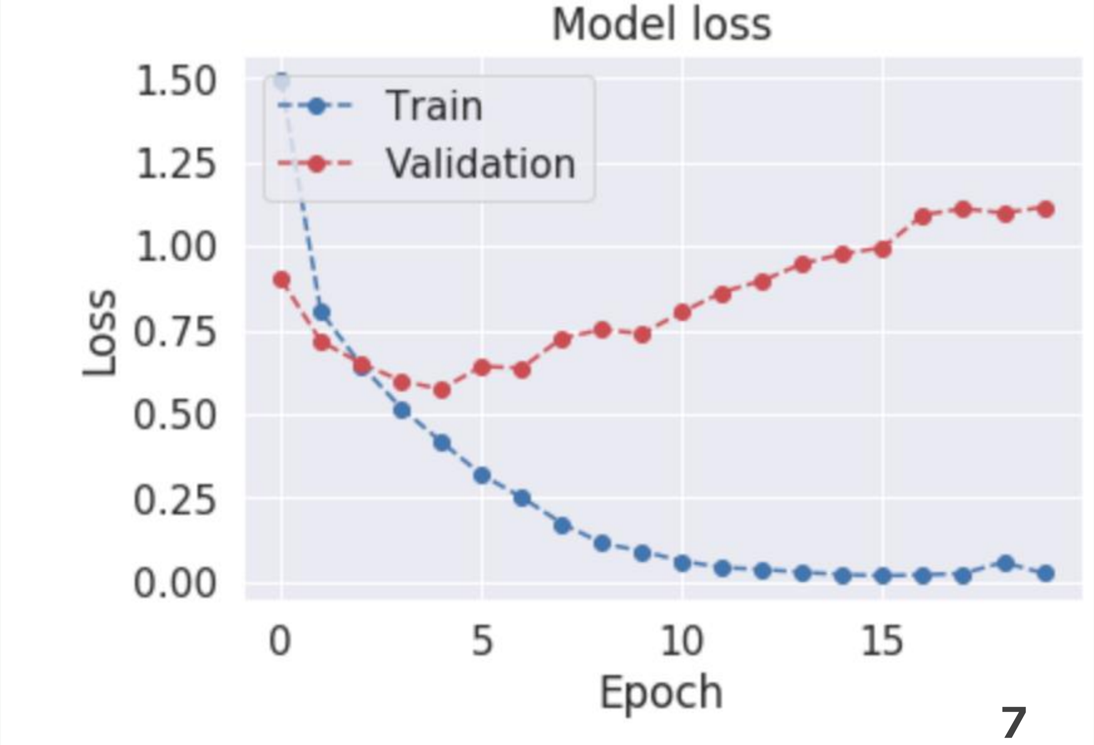
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 41472)             0
_________________________________________________________________
dense (Dense)                (None, 128)               5308544
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 5,319,462
Trainable params: 5,319,462
Non-trainable params: 0
```

Training CNN model

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
trained = model.fit(train_images,train_labels,epochs=20,batch_size=128,validation_split=0.20)
```

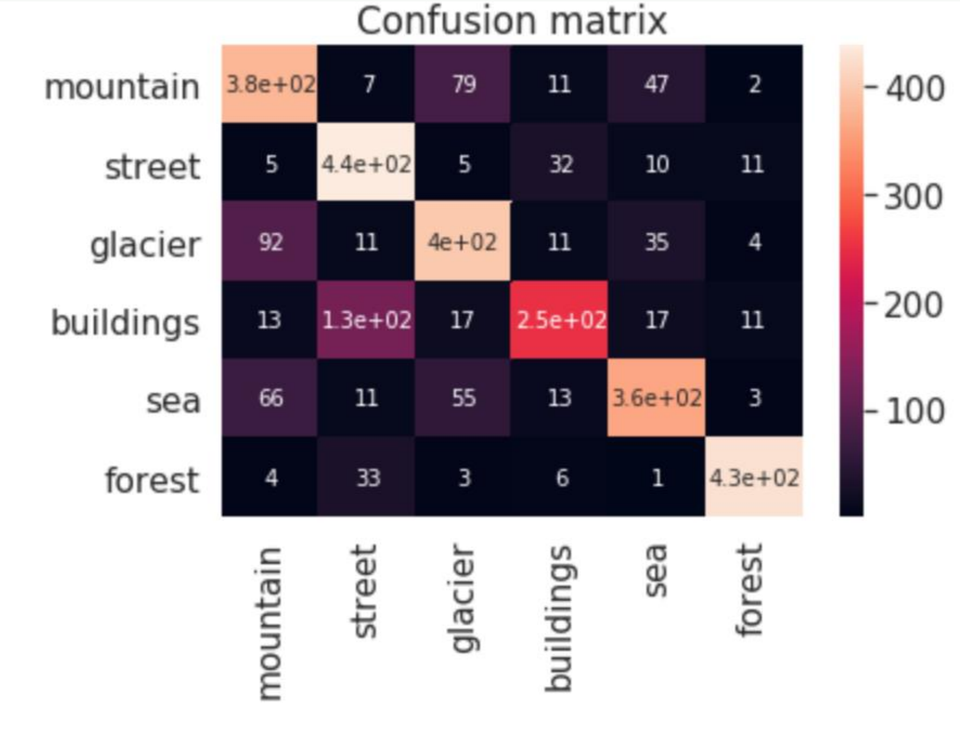




CNN model performance on test data

```
# evaluate the model on test set
test_loss = model.evaluate(test_images,test_labels, verbose=1)
```

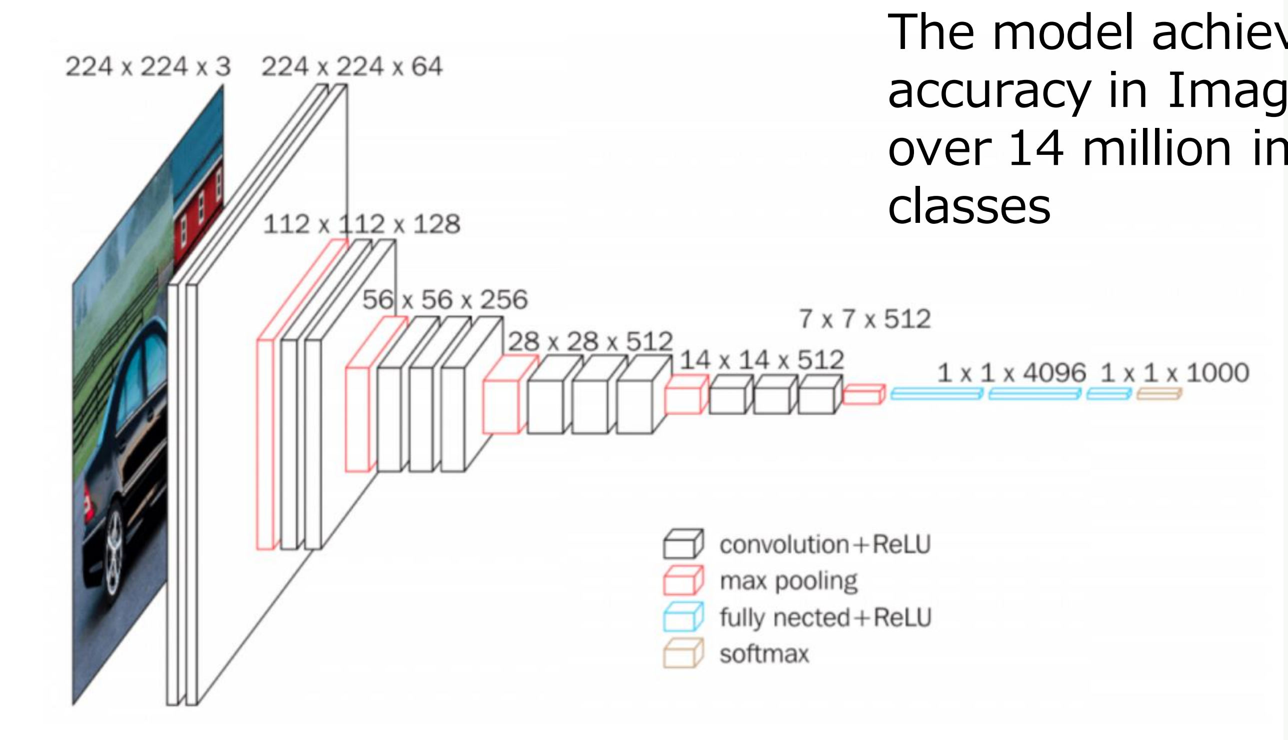
```
94/94 [==============================] - 11s 117ms/step - loss: 1.2772 - accuracy: 0.7523
```



Accuracy 75.23%

Using VGG16 pre-trained network

https://neurohive.io/en/popular-networks/vgg16/



The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes

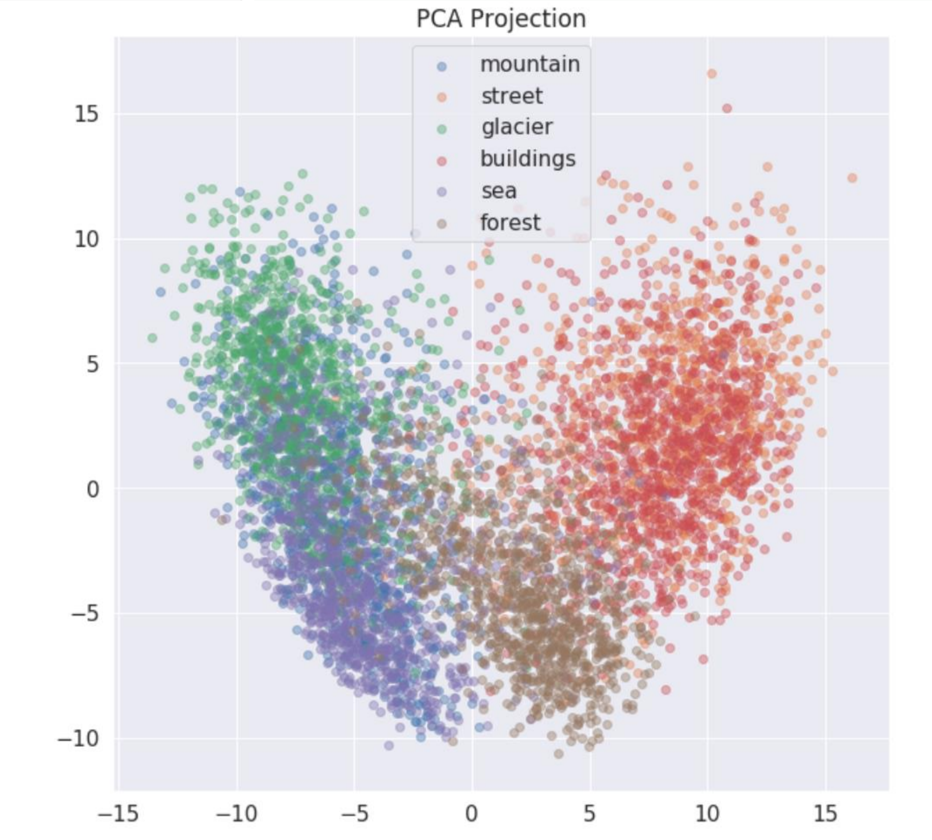
Extract features using VGG16

```
%%time

train_features = model1.predict(train_images)
test_features = model1.predict(test_images)
```

```
CPU times: user 1h 55min 5s, sys: 3min 6s, total: 1h 58min 12s
Wall time: 30min 45s
```

Principal component analysis

```python
# principal component analysis

n_train, x, y, z = train_features.shape
n_test, x, y, z = test_features.shape
numFeatures = x * y * z

from sklearn import decomposition

pca = decomposition.PCA(n_components = 2)

X = train_features.reshape((n_train, x*y*z))
pca.fit(X)

C = pca.transform(X) #
C1 = C[:,0]
C2 = C[:,1]
```



Create Simple Neural Network to Classify Extracted Features

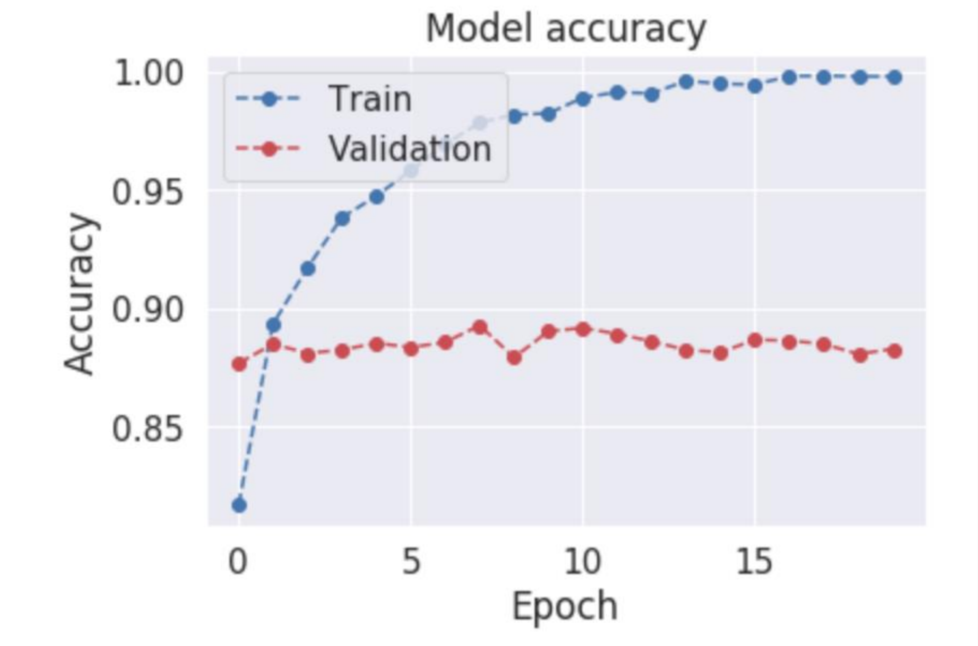
```
# create NN to predict labels

model2 = Models.Sequential()


model2.add(Layers.Flatten(input_shape = (x, y, z)))
model2.add(Layers.Dense(100,activation='relu'))
model2.add(Layers.Dense(6,activation='softmax'))

model2.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_8 (Dense)              (None, 100)               819300
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 606
=================================================================
Total params: 819,906
Trainable params: 819,906
Non-trainable params: 0
```

Train Neural Network on Extracted Features

```
model2.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# train a neural network on features extracted from VGG

trained2 = model2.fit(train_features, train_labels, batch_size=128, epochs=20, validation_split = 0.2)
```

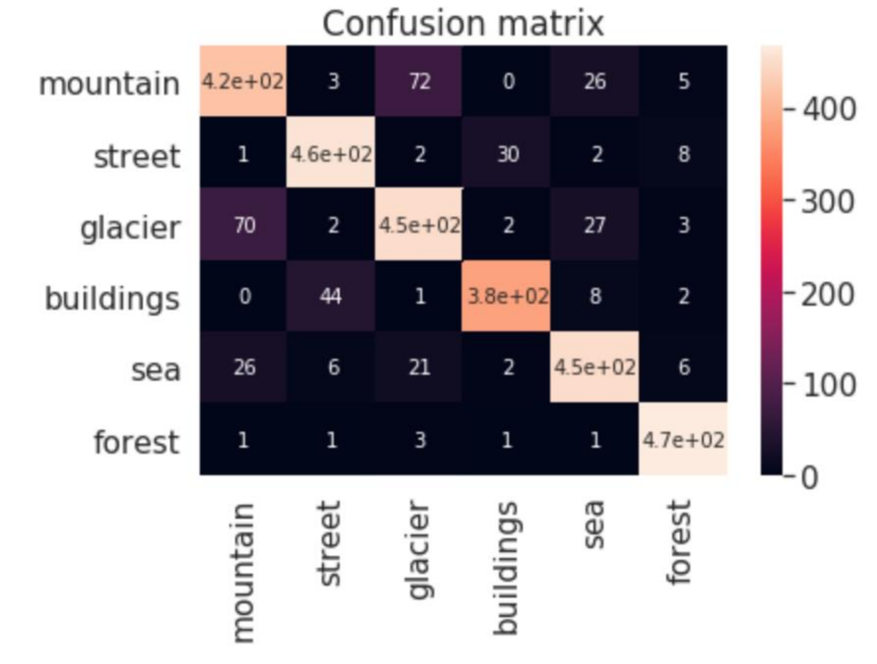


Model Performance on Test Data

```
# evaluate the model on test set
test_loss2 = model2.evaluate(test_features,test_labels, verbose=1)
```

```
94/94 [==============================] - 0s 3ms/step - loss: 0.5301 - accuracy: 0.8747
```

Accuracy 87.47%



Comparison of two models

	CNN	VGG16-based
Accuracy	**75.23%**	**87.47%**
Confusion matrix