

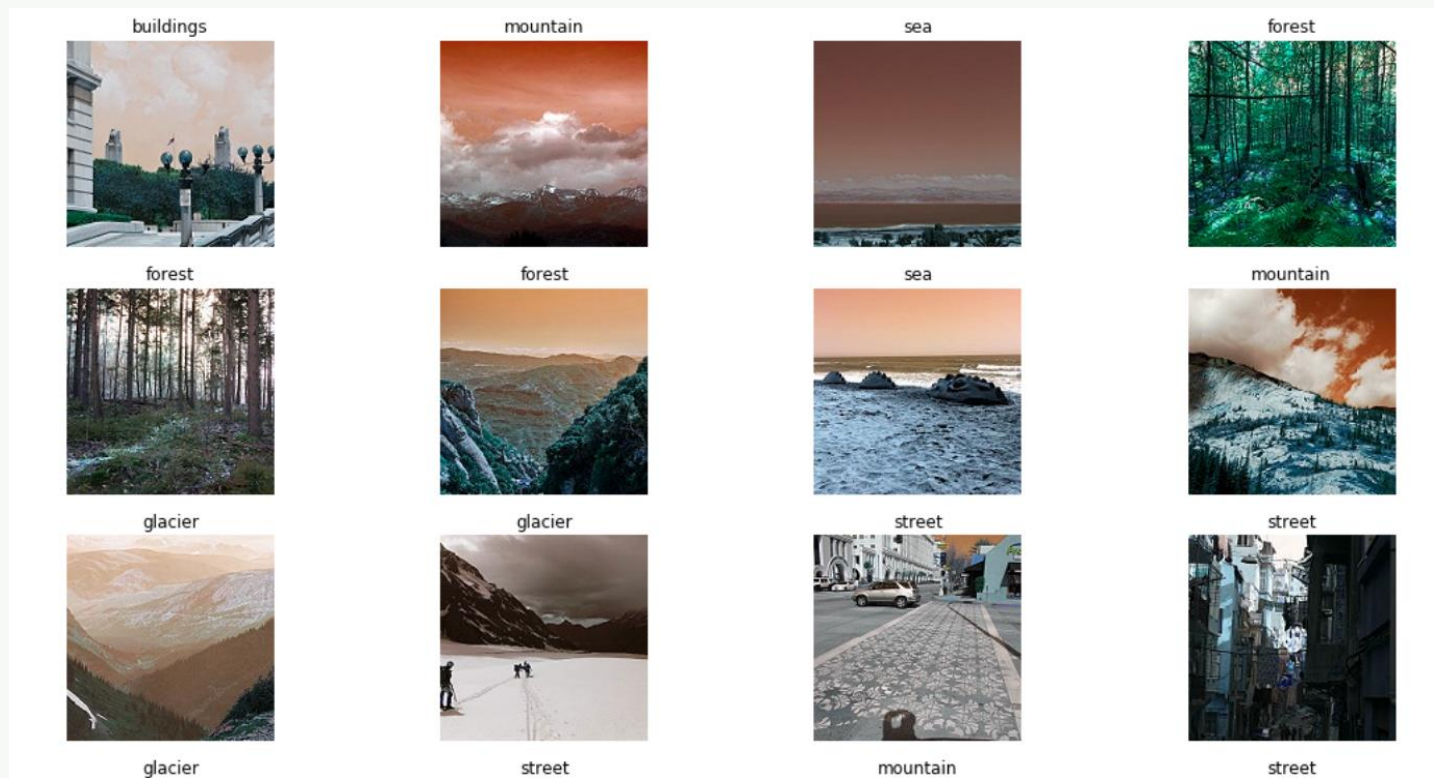
# IBM ADVANCED DATA SCIENCE CAPSTONE PROJECT

Yaroslav Aulin

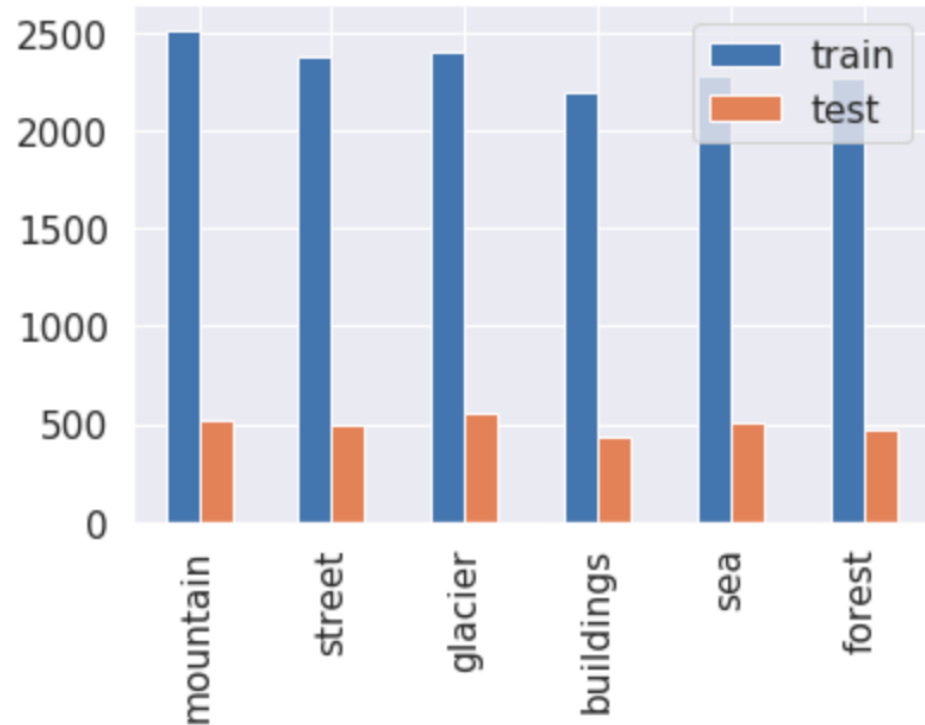
```
mirror_mod = modifier_ob.  
Set mirror object to mirror.  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
-- OPERATOR CLASSES ----  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
context):  
context.active_object is not
```

# Dataset

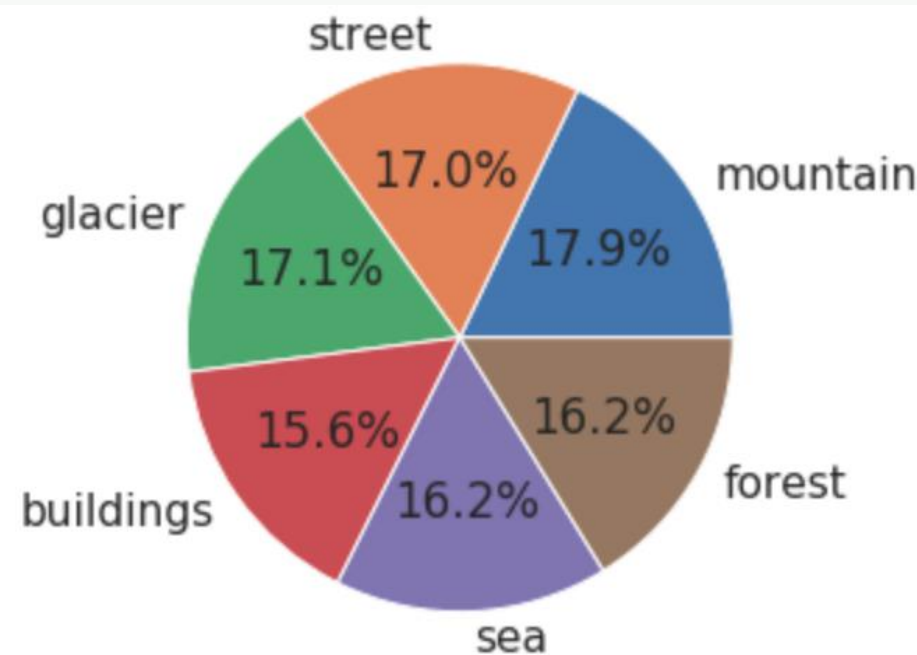
<https://www.kaggle.com/puneet6060/intel-image-classification>



# Exploring Dataset



Train Set

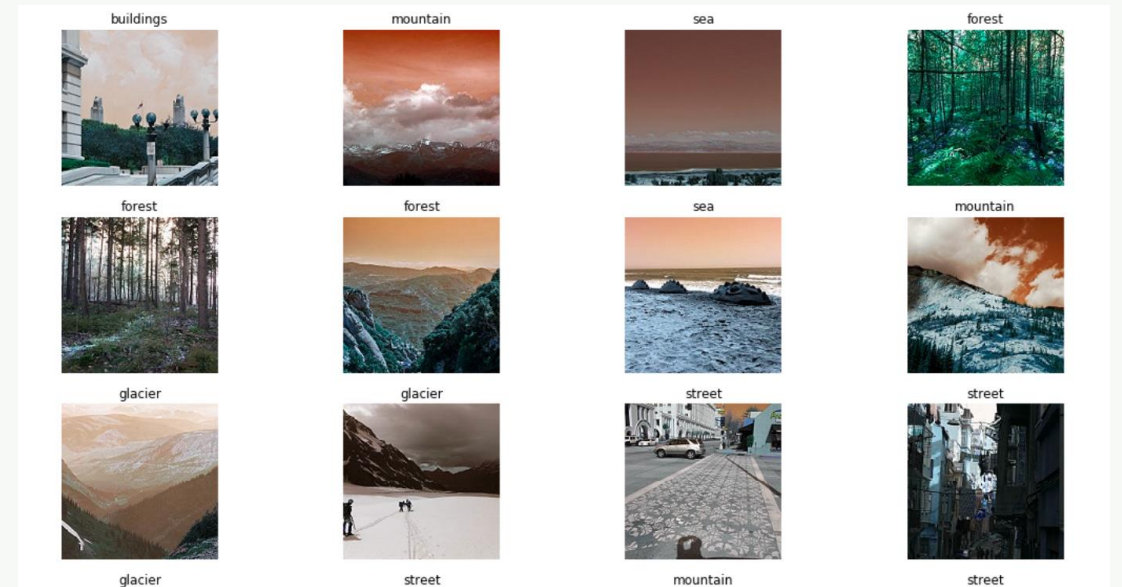


# Data Preprocessing

**Make all images the same size:**

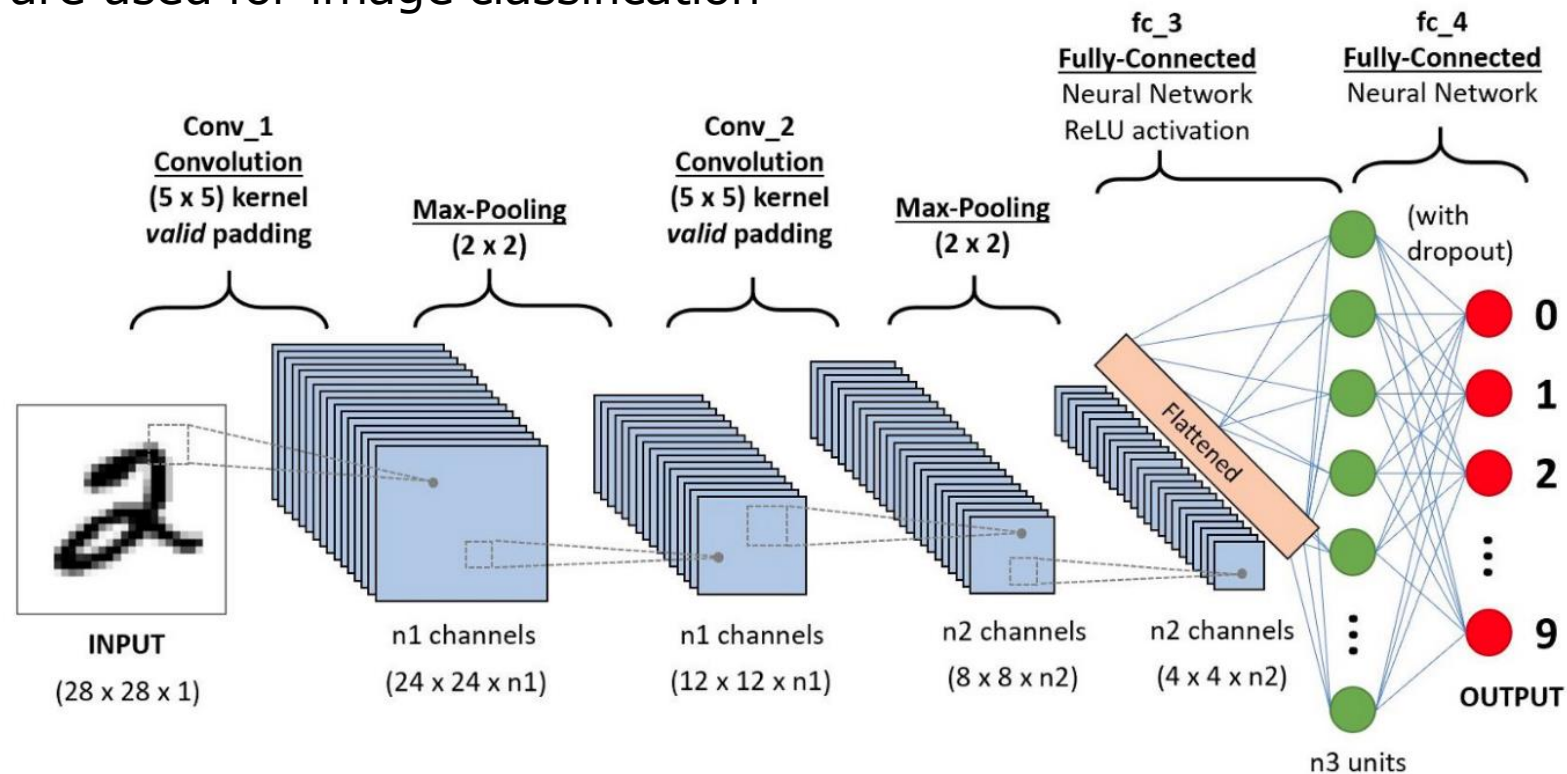
`IMAGE_SIZE = (150, 150)`

`image = cv2.resize(image, IMAGE_SIZE)`



# Convolutional neural network

CNNs are used for image classification





# 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"

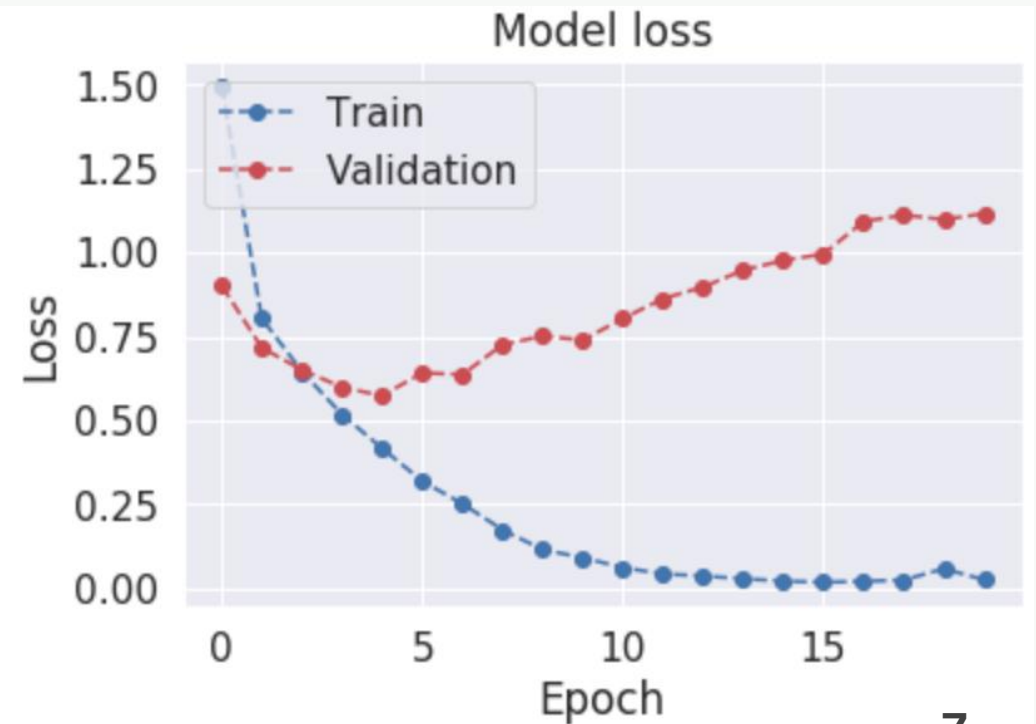
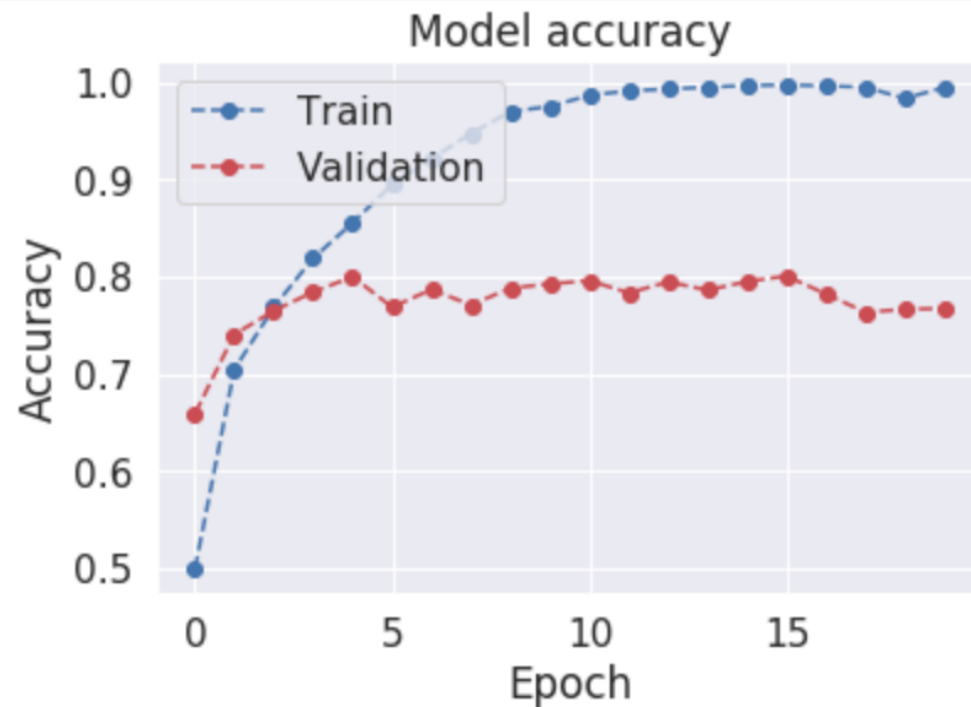
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 148, 148, 32)	896
-----		
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
-----		
conv2d_1 (Conv2D)	(None, 72, 72, 32)	9248
-----		
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
-----		
flatten (Flatten)	(None, 41472)	0
-----		
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)
```

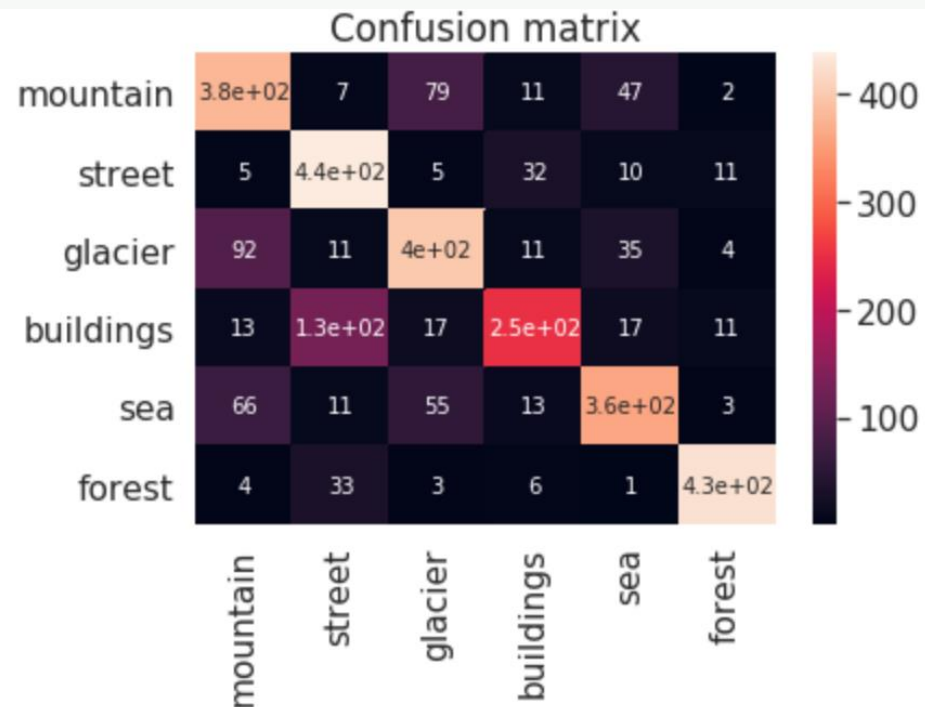


# CNN model performance on test data

```
# 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

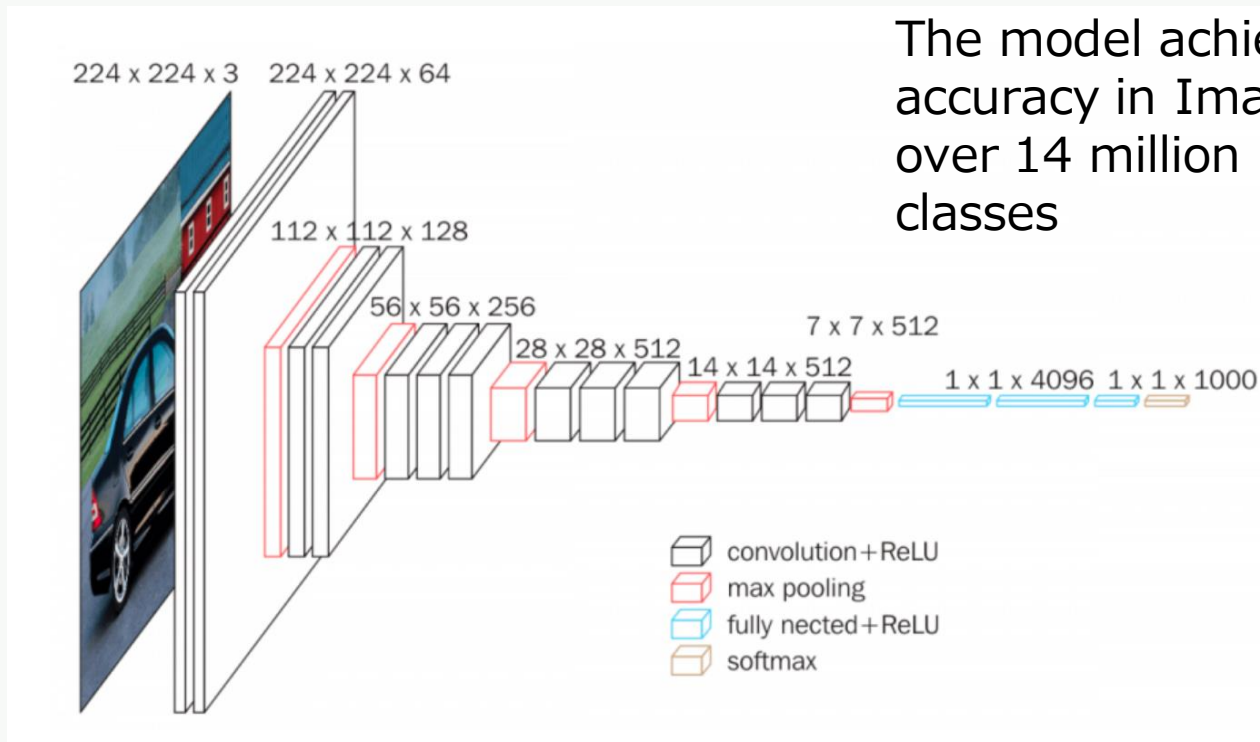
**Accuracy 75.23%**





# 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

```
: # principal component analysis

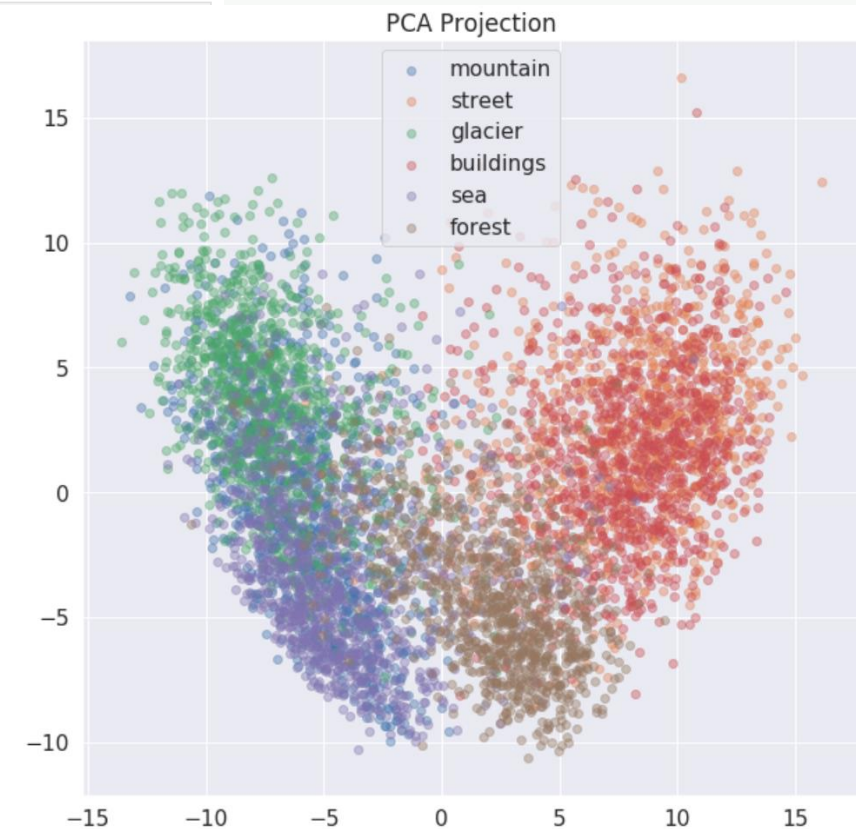
n_train, x, y, z = train_features.shape
n_test, x, y, z = test_features.shape
numFeatures = x * y * z

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"

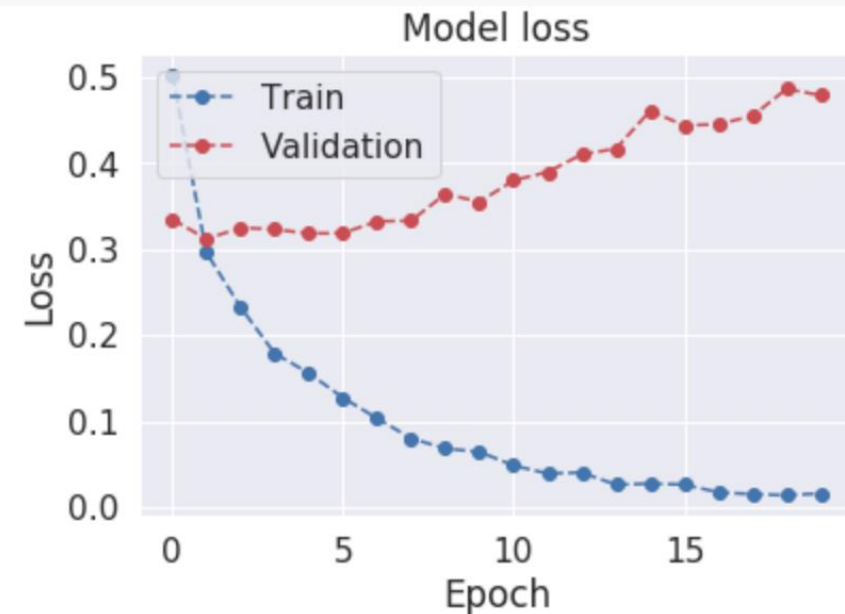
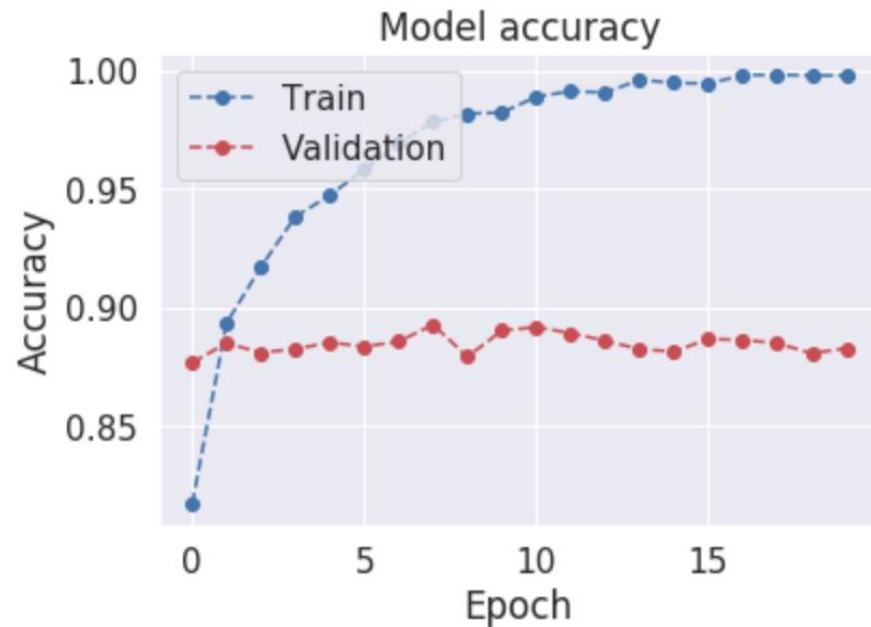
Layer (type)	Output Shape	Param #
=====		
flatten_4 (Flatten)	(None, 8192)	0
dense_8 (Dense)	(None, 100)	819300
dense_9 (Dense)	(None, 6)	606
=====		
Total params: 819,906		
Trainable params: 819,906		
Non-trainable params: 0		

# 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)
```

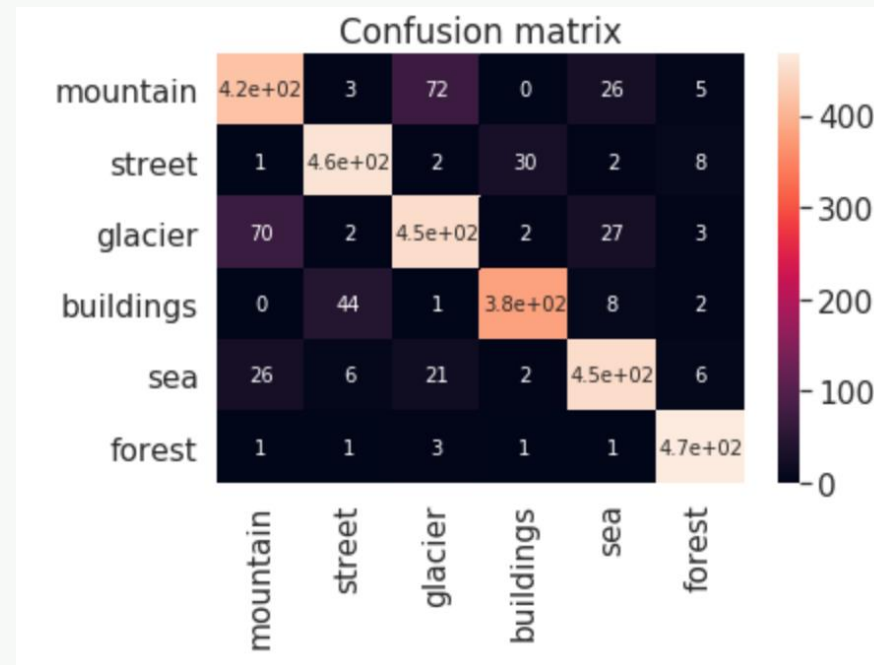


# Model Performance on Test Data

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

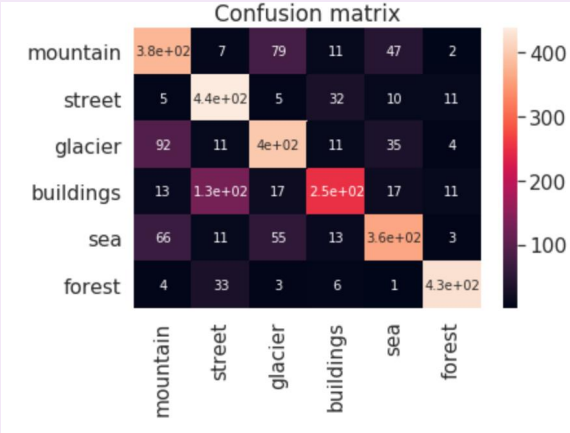
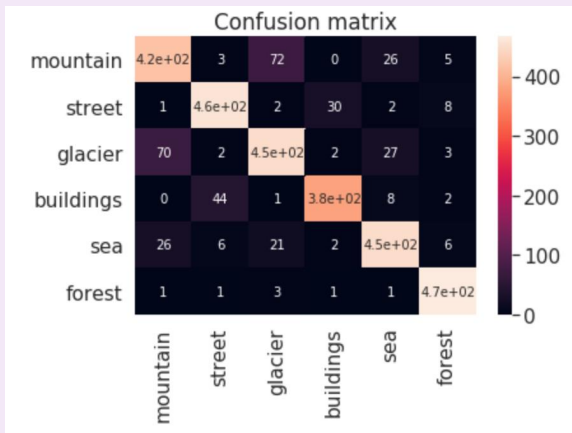
94/94 [=====] - 0s 3ms/step - loss: 0.5301 - accuracy: 0.8747

**Accuracy 87.47%**





# Comparison of two models

	CNN	VGG16-based																																																																																																		
Accuracy	75.23%	87.47%																																																																																																		
Confusion matrix	<div><p>Confusion matrix</p><table><tr><th></th><th>mountain</th><th>street</th><th>glacier</th><th>buildings</th><th>sea</th><th>forest</th></tr><tr><th>mountain</th><td>3.8e+02</td><td>7</td><td>79</td><td>11</td><td>47</td><td>2</td></tr><tr><th>street</th><td>5</td><td>4.4e+02</td><td>5</td><td>32</td><td>10</td><td>11</td></tr><tr><th>glacier</th><td>92</td><td>11</td><td>4e+02</td><td>11</td><td>35</td><td>4</td></tr><tr><th>buildings</th><td>13</td><td>1.3e+02</td><td>17</td><td>2.5e+02</td><td>17</td><td>11</td></tr><tr><th>sea</th><td>66</td><td>11</td><td>55</td><td>13</td><td>3.6e+02</td><td>3</td></tr><tr><th>forest</th><td>4</td><td>33</td><td>3</td><td>6</td><td>1</td><td>4.3e+02</td></tr></table></div>		mountain	street	glacier	buildings	sea	forest	mountain	3.8e+02	7	79	11	47	2	street	5	4.4e+02	5	32	10	11	glacier	92	11	4e+02	11	35	4	buildings	13	1.3e+02	17	2.5e+02	17	11	sea	66	11	55	13	3.6e+02	3	forest	4	33	3	6	1	4.3e+02	<div><p>Confusion matrix</p><table><tr><th></th><th>mountain</th><th>street</th><th>glacier</th><th>buildings</th><th>sea</th><th>forest</th></tr><tr><th>mountain</th><td>4.2e+02</td><td>3</td><td>72</td><td>0</td><td>26</td><td>5</td></tr><tr><th>street</th><td>1</td><td>4.6e+02</td><td>2</td><td>30</td><td>2</td><td>8</td></tr><tr><th>glacier</th><td>70</td><td>2</td><td>4.5e+02</td><td>2</td><td>27</td><td>3</td></tr><tr><th>buildings</th><td>0</td><td>44</td><td>1</td><td>3.8e+02</td><td>8</td><td>2</td></tr><tr><th>sea</th><td>26</td><td>6</td><td>21</td><td>2</td><td>4.5e+02</td><td>6</td></tr><tr><th>forest</th><td>1</td><td>1</td><td>3</td><td>1</td><td>1</td><td>4.7e+02</td></tr></table></div>		mountain	street	glacier	buildings	sea	forest	mountain	4.2e+02	3	72	0	26	5	street	1	4.6e+02	2	30	2	8	glacier	70	2	4.5e+02	2	27	3	buildings	0	44	1	3.8e+02	8	2	sea	26	6	21	2	4.5e+02	6	forest	1	1	3	1	1	4.7e+02
	mountain	street	glacier	buildings	sea	forest																																																																																														
mountain	3.8e+02	7	79	11	47	2																																																																																														
street	5	4.4e+02	5	32	10	11																																																																																														
glacier	92	11	4e+02	11	35	4																																																																																														
buildings	13	1.3e+02	17	2.5e+02	17	11																																																																																														
sea	66	11	55	13	3.6e+02	3																																																																																														
forest	4	33	3	6	1	4.3e+02																																																																																														
	mountain	street	glacier	buildings	sea	forest																																																																																														
mountain	4.2e+02	3	72	0	26	5																																																																																														
street	1	4.6e+02	2	30	2	8																																																																																														
glacier	70	2	4.5e+02	2	27	3																																																																																														
buildings	0	44	1	3.8e+02	8	2																																																																																														
sea	26	6	21	2	4.5e+02	6																																																																																														
forest	1	1	3	1	1	4.7e+02																																																																																														

15