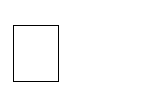
**Arya Teli**

**2213138**

**AIA-1-A**

**Assignment No. 3**

**Aim:** Develop classification model for cat-dogs dataset using CNN model. Analyze the model accuracy and generate classification report.

 Develop an application and test the user given inputs.

Analyze the result with and without regularization/dropout

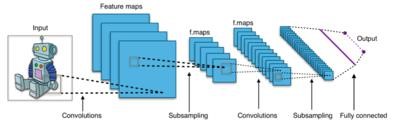
**Objectives:**

1. To learn about classification
2. To learn CNN
3. To demonstrate and analyse the results

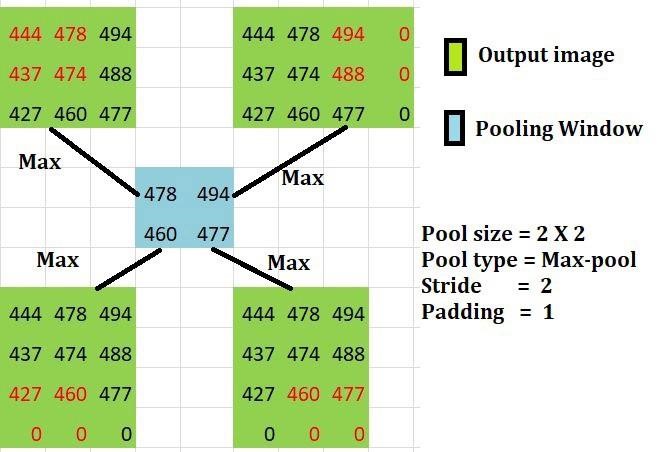
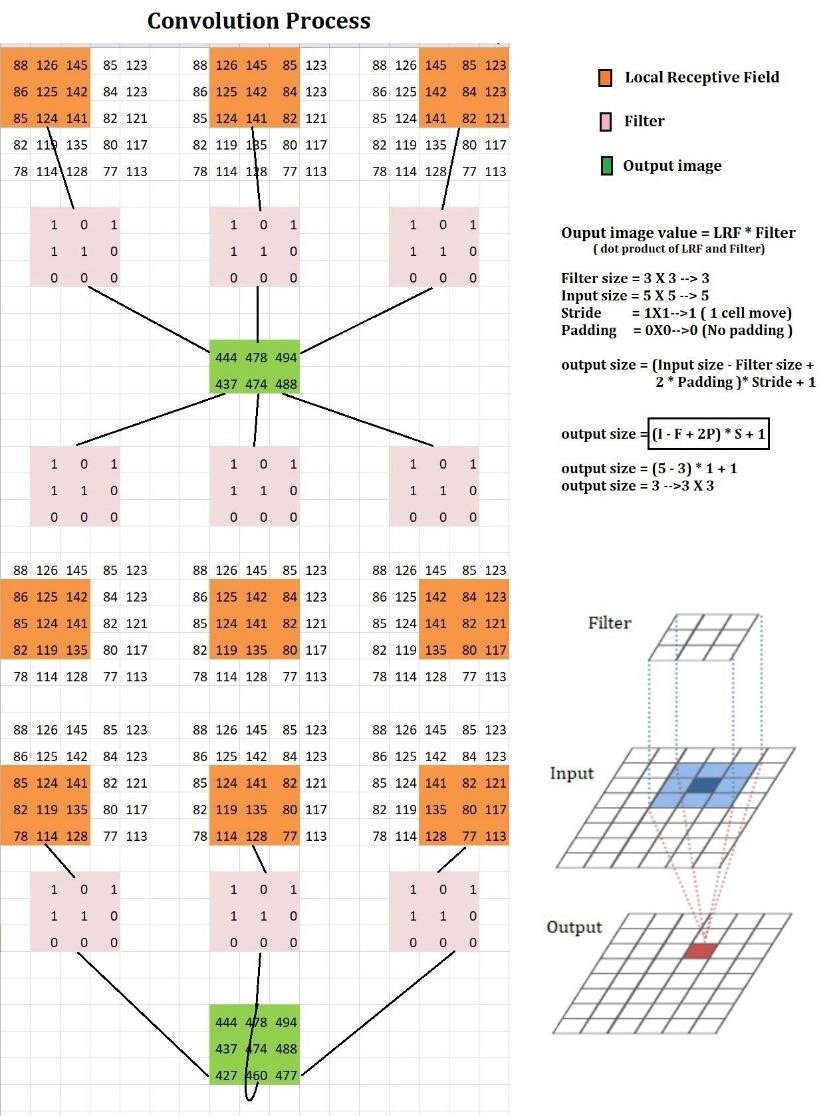
**Theory:**

A **convolutional neural network** (**CNN**, or **ConvNet**) is a class of [deep neural networks,](https://en.wikipedia.org/wiki/Deep_neural_network) most commonly applied to analysing visual imagery. They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and [translation invariance](https://en.wikipedia.org/wiki/Translation_invariance) characteristics. They have applications in [image and video recognition,](https://en.wikipedia.org/wiki/Computer_vision) [recommender systems,](https://en.wikipedia.org/wiki/Recommender_system) [image classification,](https://en.wikipedia.org/wiki/Image_classification) [medical image analysis,](https://en.wikipedia.org/wiki/Medical_image_computing) [natural language processing,](https://en.wikipedia.org/wiki/Natural_language_processing) [brain-computer interfaces,](https://en.wikipedia.org/wiki/Brain%E2%80%93computer_interface) and financial [time series.](https://en.wikipedia.org/wiki/Time_series)

CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptron.](https://en.wikipedia.org/wiki/Multilayer_perceptron) Multilayer perceptron usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.



**Mathematics**



**Batch Normalization**

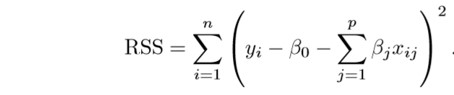
**Batch normalization** is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-**batch**. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

# Regularisation

This technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. A simple relation for linear regression looks like this. Here Y represents the learned relation and β represents the coefficient estimates for different variables or predictors(X).

Y ≈ β0 + β1X1 + β2X2 + …+ βpXp

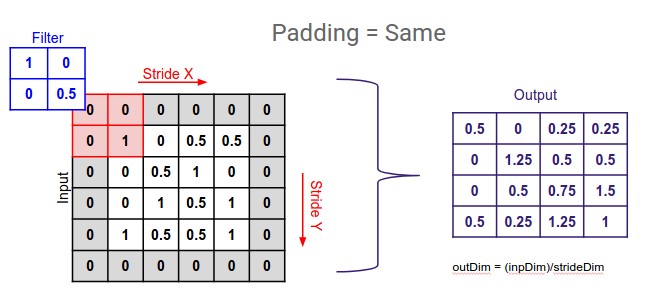
The fitting procedure involves a loss function, known as residual sum of squares or RSS. The coefficients are chosen, such that they minimize this loss function.



Now, this will adjust the coefficients based on your training data. If there is noise in the training data, then the estimated coefficients won’t generalize well to the future data. This is where regularization comes in and shrinks or regularizes these learned estimates towards zero.

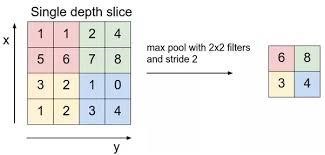
**Padding**

**Padding** is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a **CNN**. For example, if the **padding** in a **CNN** is set to zero, then every pixel value that is added will be of value zero.



**Strides**

**Stride** is the number of pixels shifts over the input matrix. When the **stride** is 1 then we move the filters to 1 pixel at a time. When the **stride** is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a **stride** of 2.



**Code:**

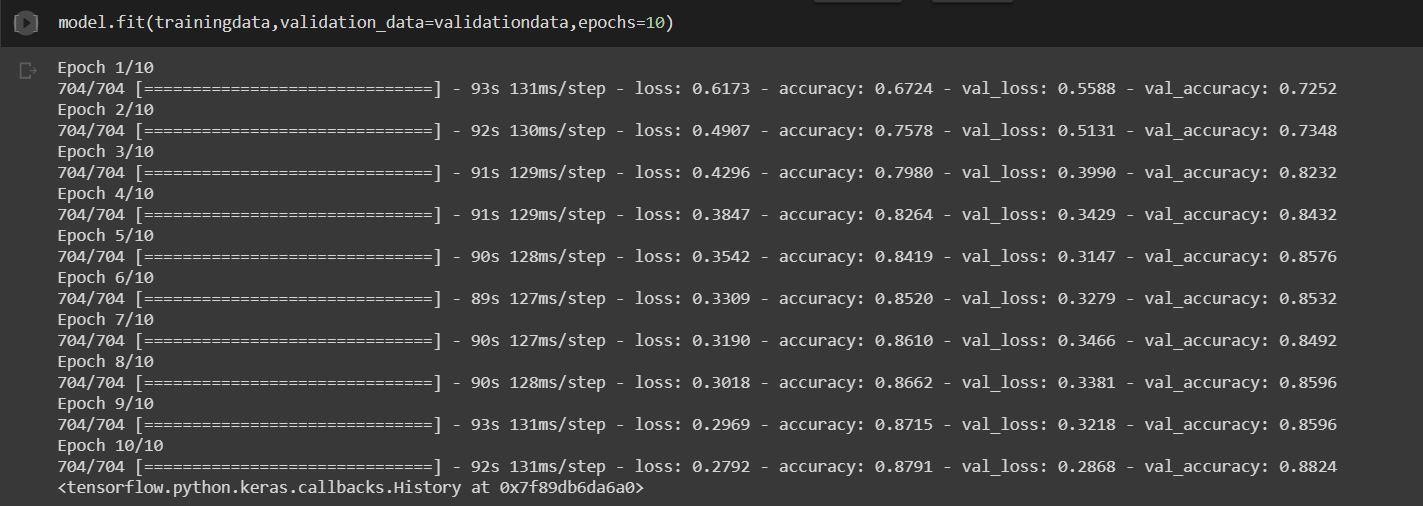
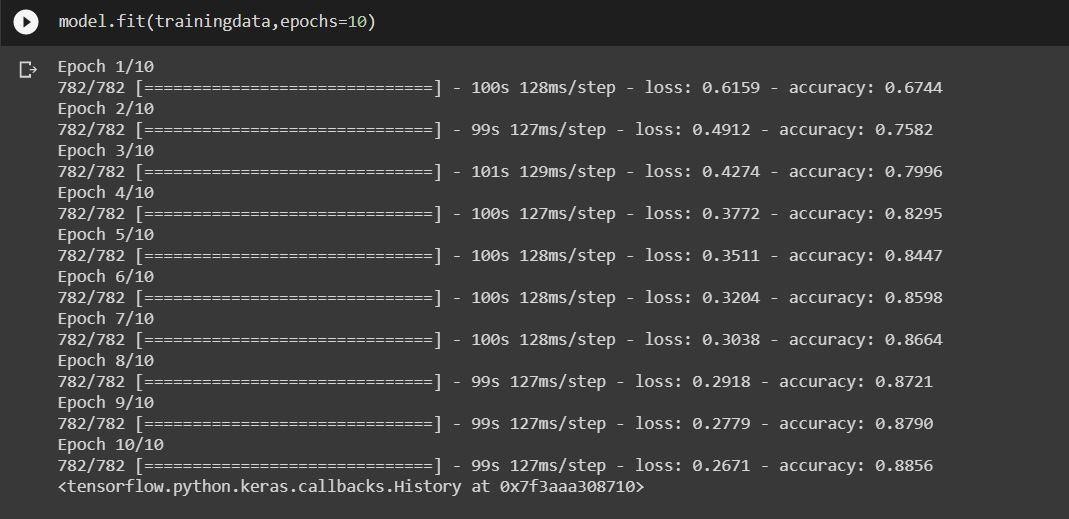
|  |
| --- |
| from pydrive.auth import GoogleAuth from pydrive.drive import GoogleDrive from google.colab import auth from oauth2client.client import GoogleCredentials  auth.authenticate\_user() gauth = GoogleAuth() gauth.credentials = GoogleCredentials.get\_application\_default() drive = GoogleDrive(gauth)  downloaded = drive.CreateFile({'id':"1NwrithRqi1lcpAPadM5Rz0AQrqmr-2DD"}) downloaded.GetContentFile('CatvsDogs.rar')    !unrar x "/content/CatvsDogs.rar" "/content/"  import pandas as pd import tensorflow as tf from tensorflow.keras import models,Sequential,layers,preprocessing import os  file\_names=os.listdir("/content/train") dogorcat=[] for name in file\_names:  category=name.split('.')[0] if category=='dog':  dogorcat.append("DOG") else:  dogorcat.append("CAT") train=pd.DataFrame({ 'filename':file\_names,  'category':dogorcat  }) model=models.Sequential() model.add(layers.BatchNormalization()) model.add(layers.Conv2D(32,(3,3),activation='relu',input\_shape=(64,64,3)))  #model.add(layers.Conv2D(32,(3,3),activation='relu')) model.add(layers.BatchNormalization()) |
| model.add(layers.MaxPooling2D(2,2)) model.add(layers.Conv2D(64,(3,3),activation='relu')) model.add(layers.BatchNormalization())  #model.add(layers.Conv2D(64,(3,3),activation='relu')) model.add(layers.MaxPooling2D(2,2)) model.add(layers.Conv2D(128,(3,3),activation='relu')) model.add(layers.BatchNormalization())  #model.add(layers.Conv2D(128,(3,3),activation='relu')) model.add(layers.MaxPooling2D(2,2)) model.add(layers.Flatten()) model.add(layers.Dense(512,activation='relu')) model.add(layers.Dense(256,activation="relu")) model.add(layers.Dense(128,activation="relu")) model.add(layers.Dense(2,activation="softmax"))  model.compile(optimizer="adam",loss='categorical\_crossentropy',metrics=['accur acy']) training = preprocessing.image.ImageDataGenerator(rotation\_range=15, rescale=1 ./255, shear\_range=0.1, zoom\_range=0.2, horizontal\_flip=True, width\_shift\_rang e=0.1, height\_shift\_range=0.1) validation=preprocessing.image.ImageDataGenerator(rotation\_range=15, rescale=1 ./255, shear\_range=0.1, zoom\_range=0.2, horizontal\_flip=True, width\_shift\_rang e=0.1, height\_shift\_range=0.1)  trainingdata = training.flow\_from\_dataframe(train,"/content/train",x\_col='file name',y\_col='category',target\_size=(64,64),class\_mode='categorical')  model.fit(trainingdata,epochs=10)  from google.colab import drive drive.mount('/content/drive')  os.makedirs("drive/My Drive/Models",exist\_ok=True) model.save("drive/My Drive/Models") |

**GUI**

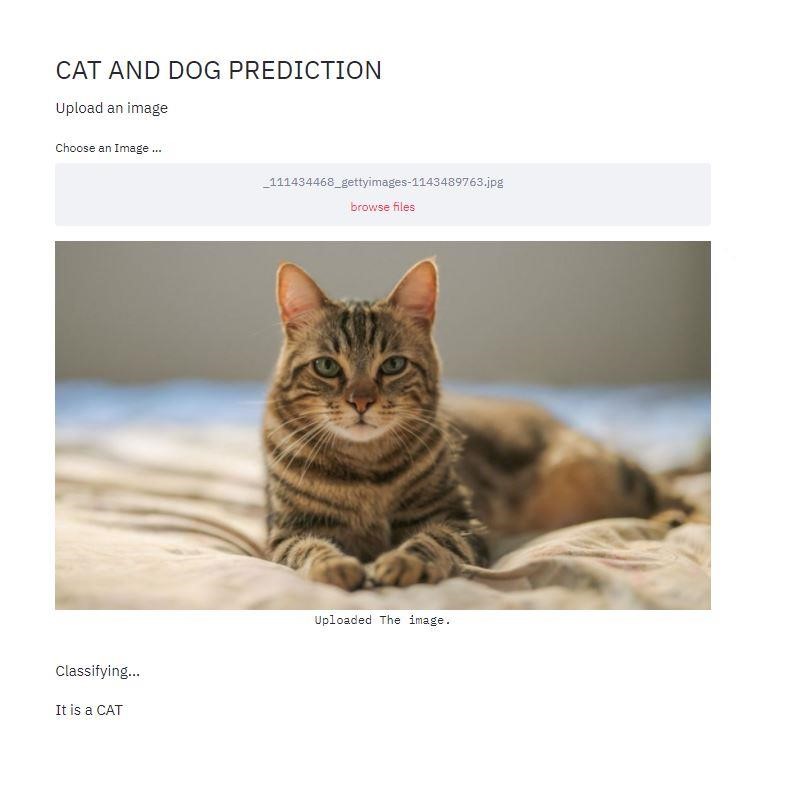
|  |
| --- |
| from PIL import Image,ImageOps import os import streamlit as st import tensorflow as tf  from tensorflow.keras import models  from tensorflow.keras.preprocessing import image import numpy as np st.set\_option('deprecation.showfileUploaderEncoding', False) def load\_models(img):  model = models.load\_model('D:/Programs/ML1/ML2/Assignment3/') |

|  |
| --- |
| image=img.resize((64,64)) image\_array=np.array(image)  #image\_array=tf.image.rgb\_to\_grayscale(image\_array) image\_array=(tf.reshape(image\_array,(image\_array.shape[0],image\_array.shap e[0],3)))/255 image\_array=np.array([image\_array])  prediction = model.predict\_classes(image\_array) return prediction    def upload\_images():  uploaded\_file = st.file\_uploader("Choose an Image ...", type="jpg") if uploaded\_file is not None:  image = Image.open(uploaded\_file) st.image(image, caption='Uploaded The image.', use\_column\_width=True) st.write("") st.write("Classifying...") label = load\_models(image)  if label==0:  st.write("It is a CAT") if label==1:  st.write("It is a Dog")            if \_\_name\_\_ =="\_\_main\_\_":  st.header("CAT AND DOG PREDICTION") st.write("Upload an image")  upload\_images() |

**Results:**



GUI



**Conclusion:**

Thus, we have understood how and where CNN is used and how it is programmed in TensorFlow.

We have also been able to polish GUI creation skills even further.