S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 07

Aim: Perform and implement Convolution Neural Network for diabetic dataset.

Name of Student	**: Shrutika Pradeep Bagdi**
Roll No	**: CS22130**
Semester/Year	**: VIIth Sem / IVth Year**
Academic Session	**: 2025-2026 [ODD]**
Date of Performance	: ______________________________
Date of Submission	: ______________________________

Aim: Perform and implement Convolution Neural Network for diabetic dataset.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

It is computationally ineffective right. So here comes Convolutional Neural Network or CNN. In simple word what CNN does is, it extract the feature of image and convert it into lower dimension without loosing its characteristics. In the following example you can see that initial the size of the image is 224 x 224 x

HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirement: Computer system with high configurations

Software Requirement: Google Colab

THEORY:

An introductory look at Convolutional Neural Network with theory and code example. I want to write about one of the most important neural networks used in the field of deep learning, especially for image recognition and natural language processing: convolutional neural network, also called "CNN" or "ConvNet".

What is Convolutional neural network?

Convolutional neural networks. Sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

Architecture of a Traditional CNN

A convolutional neural network is composed of at least 3 layers:

- A **convolution layer** to perform convolution operations and to generate many feature maps from one image;
- A **pooling layer** to denoise the feature maps by shrinking non-overlapping submatrices into summary

statistics (such as maximums).

- A **dense layer** which is a usual (shallow/deep) neural network that takes flattened inputs.

Working of Convolutional Neural Networks.

Convolutional neural networks are based on neuroscience findings. They are made of layers of artificial neurons called nodes. These nodes are functions that calculate the weighted sum of the inputs and return an activation map. This is the convolution part of the neural network.

Each node in a layer is defined by its weight values. When you give a layer some data, like an image, it takes the pixel values and picks out some of the visual features.

When you're working with data in a CNN, each layer returns activation maps. These maps point out important features in the data set. If you gave the CNN an image, it'll point out features based on pixel values, like colors, and give you an activation function.

Usually with images, a CNN will initially find the edges of the picture. Then this slight definition of the image will get passed to the next layer. Then that layer will start detecting things like corners and color groups. Then that image definition will get passed to the next layer and the cycle continues until a prediction is made.

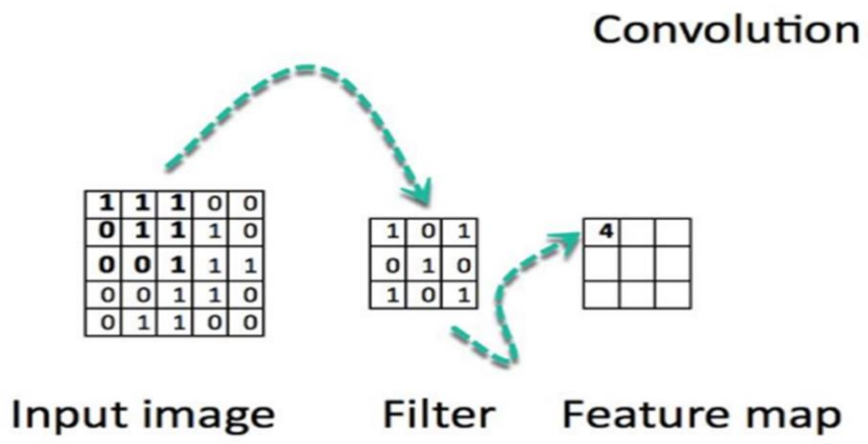
As the layers get more defined, this is called max pooling. It only returns the most relevant features from the layer in the activation map. This is what gets passed to each successive layer until you get the final layer. The last layer of a CNN is the classification layer which determines the predicted value based on the activation map. If you pass a handwriting sample to a CNN, the classification layer will tell you what letter is in the image. This is what autonomous vehicles use to determine whether an object is another car, a person, or some other obstacle.

Training a CNN is similar to training many other machine learning algorithms. You'll start with some training data that is separate from your test data and you'll tune your weights based on the accuracy of the predicted values. Just be careful that you don't overfit your model.

Different types of CNN

- **1D CNN**: With these, the CNN kernel moves in one direction. 1D CNNs are usually used on time-series data.
- **2D CNN**: These kinds of CNN kernels move in two directions. You'll see these used with image labelling and processing.
- **3D CNN**: This kind of CNN has a kernel that moves in three directions. With this type of CNN,

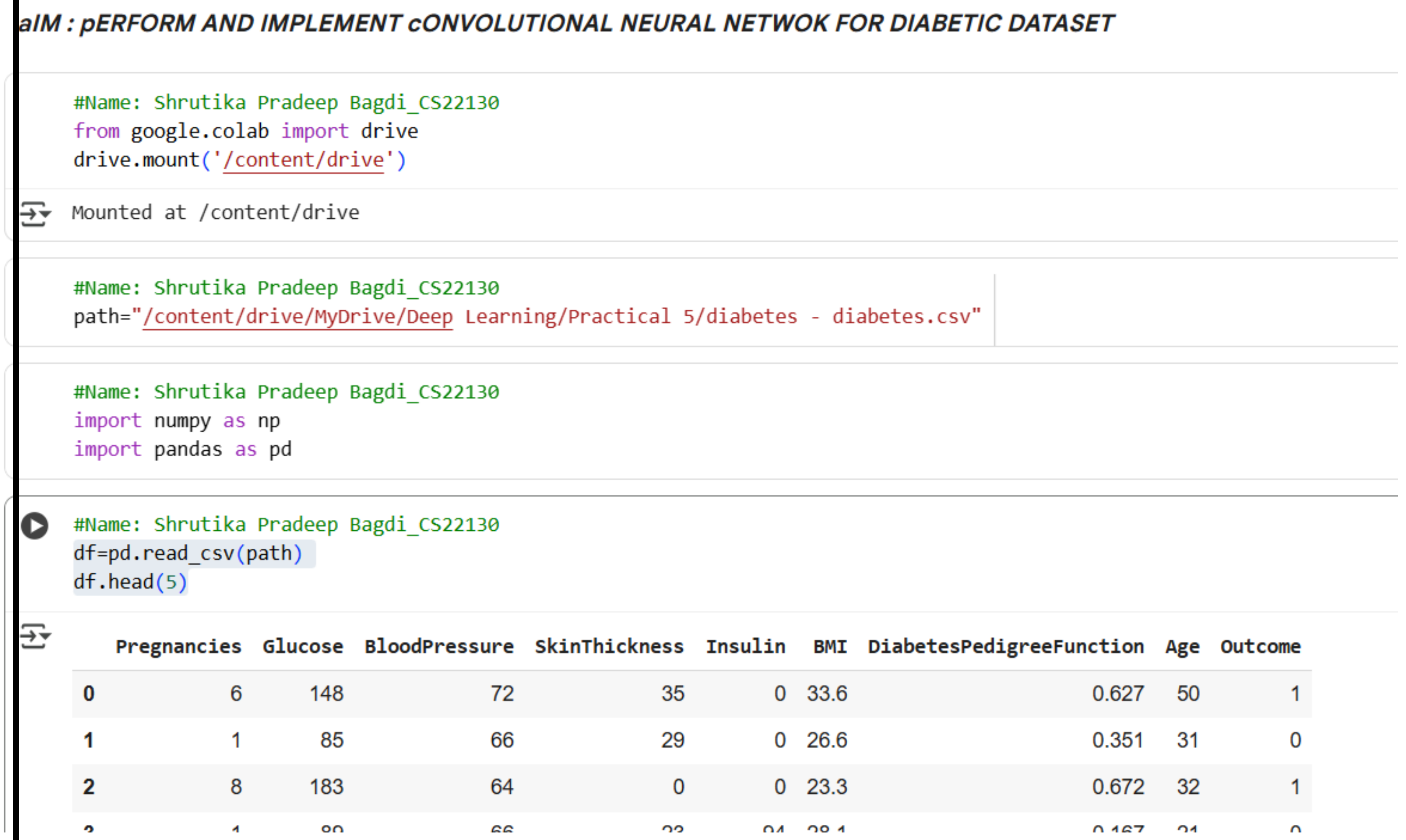
researchers use them on 3D images like CT scans and MRIs.



PROGRAM CODE:

```
from google.colab import drive
drive.mount('/content/drive')
path="/content/drive/MyDrive/Deep Learning/Practical 5/diabetes - diabetes.csv"
import numpy as np
import pandas as pd
df=pd.read_csv(path)
df.head(5)
df.shape
df.isnull().sum()
df.info()
df.describe()
df.tail()
X = df.drop('Outcome', axis=1).values
y = df['Outcome'].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
print("Train shape", X_train.shape)
print("Test shape", X_test.shape)
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
model = Sequential([
    Conv1D(32, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Conv1D(64, kernel_size=3, activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
# Compile
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
# Train
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
                    validation_data=(X_test, y_test), verbose=1)
# Evaluate
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', acc)
```

OUTPUT (SCREENSHOT):



aIM : pERFORM AND IMPLEMENT cONVOLUTIONAL NEURAL NETWOK FOR DIABETIC DATASET

```
#Name: Shrutika Pradeep Bagdi_CS22130
from google.colab import drive
drive.mount('/content/drive')
```

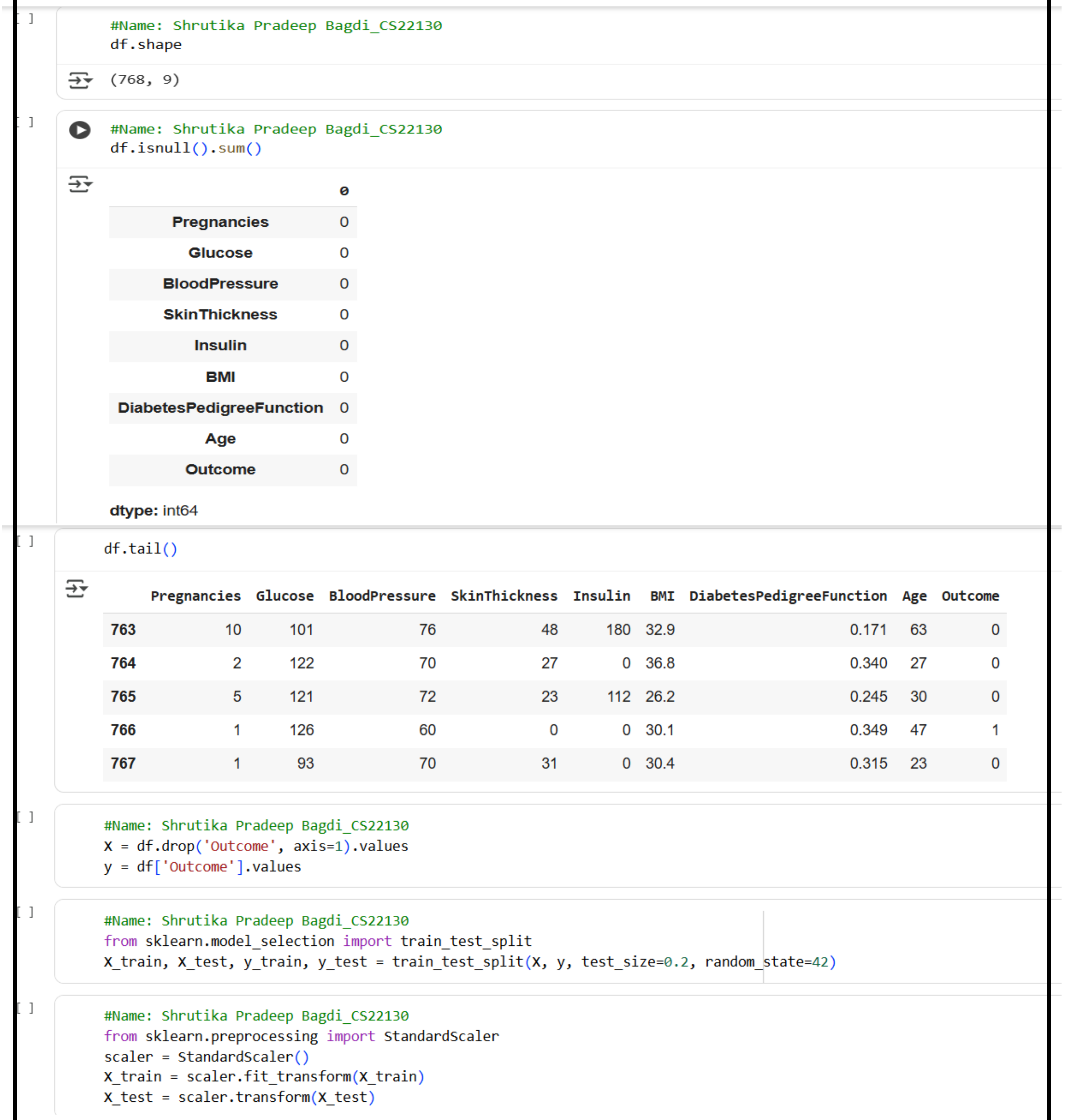
```
Mounted at /content/drive
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
path="/content/drive/MyDrive/Deep Learning/Practical 5/diabetes - diabetes.csv"
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
import numpy as np
import pandas as pd
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
df=pd.read_csv(path)
df.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0



```
#Name: Shrutika Pradeep Bagdi_CS22130
df.shape
```

```
(768, 9)
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
df.isnull().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

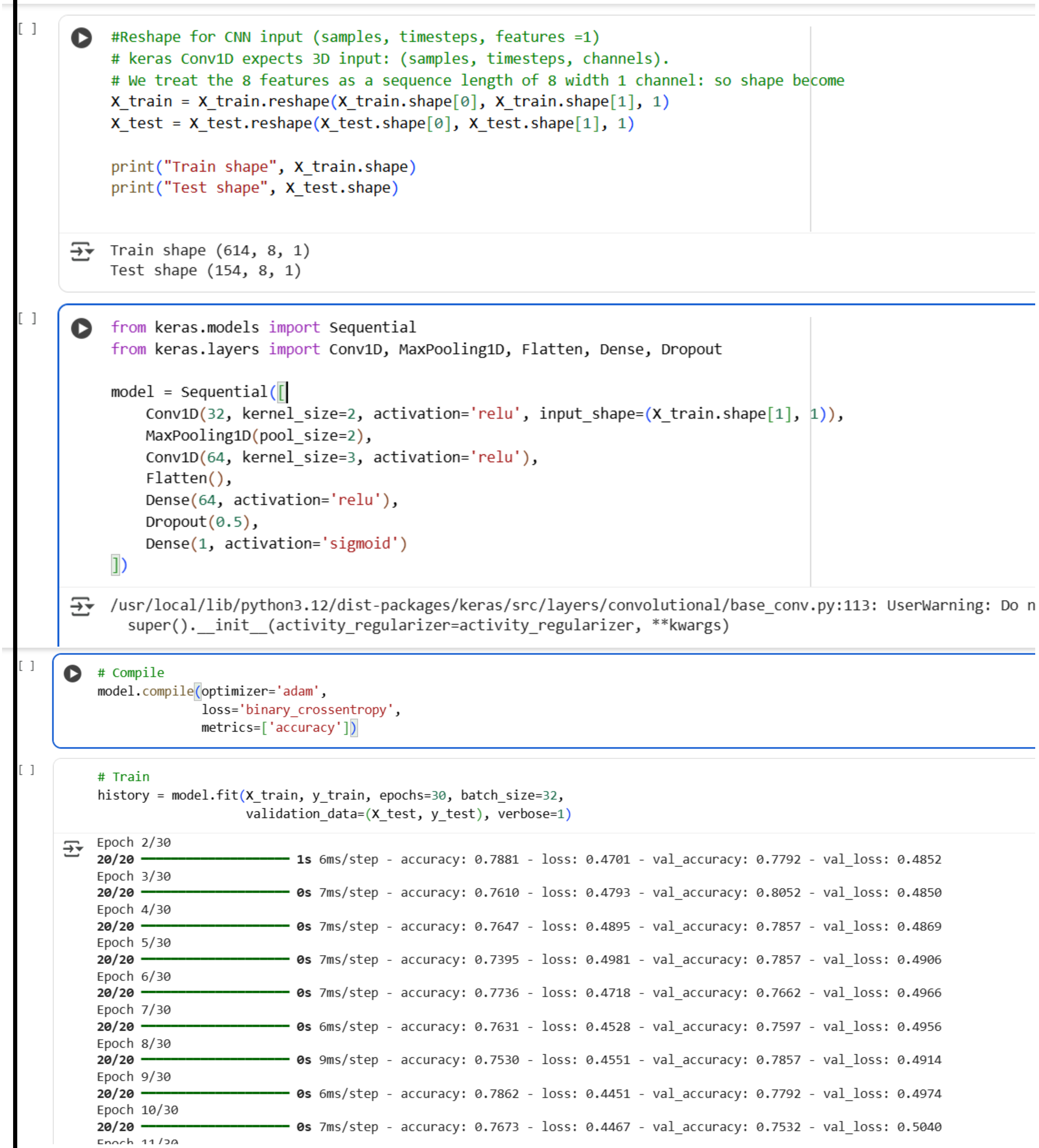
```
df.tail()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
#Name: Shrutika Pradeep Bagdi_CS22130
X = df.drop('Outcome', axis=1).values
y = df['Outcome'].values
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```



```
[ ]
#Reshape for CNN input (samples, timesteps, features =1)
# keras Conv1D expects 3D input: (samples, timesteps, channels).
# We treat the 8 features as a sequence length of 8 width 1 channel: so shape become
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

print("Train shape", X_train.shape)
print("Test shape", X_test.shape)
```

```
Train shape (614, 8, 1)
Test shape (154, 8, 1)
```

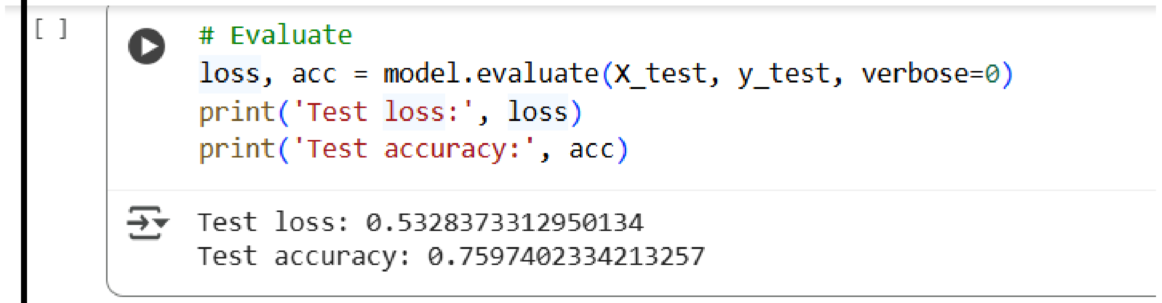
```
[ ]
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout

model = Sequential([
    Conv1D(32, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Conv1D(64, kernel_size=3, activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do n
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

```
[ ]
# Train
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
                    validation_data=(X_test, y_test), verbose=1)
```

```
Epoch 2/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7881 - loss: 0.4701 - val_accuracy: 0.7792 - val_loss: 0.4852
Epoch 3/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7610 - loss: 0.4793 - val_accuracy: 0.8052 - val_loss: 0.4850
Epoch 4/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7647 - loss: 0.4895 - val_accuracy: 0.7857 - val_loss: 0.4869
Epoch 5/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7395 - loss: 0.4981 - val_accuracy: 0.7857 - val_loss: 0.4906
Epoch 6/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7736 - loss: 0.4718 - val_accuracy: 0.7662 - val_loss: 0.4966
Epoch 7/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7631 - loss: 0.4528 - val_accuracy: 0.7597 - val_loss: 0.4956
Epoch 8/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7530 - loss: 0.4551 - val_accuracy: 0.7857 - val_loss: 0.4914
Epoch 9/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7862 - loss: 0.4451 - val_accuracy: 0.7792 - val_loss: 0.4974
Epoch 10/30
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7673 - loss: 0.4467 - val_accuracy: 0.7532 - val_loss: 0.5040
Epoch 11/30
```



```
[ ]
# Evaluate
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', acc)
```

```
Test loss: 0.5328373312950134
Test accuracy: 0.7597402334213257
```

CONCLUSION:

____

____

____

____

____

DISCUSSION AND VIVA VOCE:

1. What is CNN?
2. What are the layers of CNN?
3. What is padding in CNN?
4. How can CNN improve image classification and imge recognition?
5. Explain the different layers of CNN?

REFERENCE:

- https://towardsdatascience.com/convolutional-neural-networks-cnns-a-practical-perspective c7b3b2091aa8
- https://en.wikipedia.org/wiki/Convolutional_neural_network#References