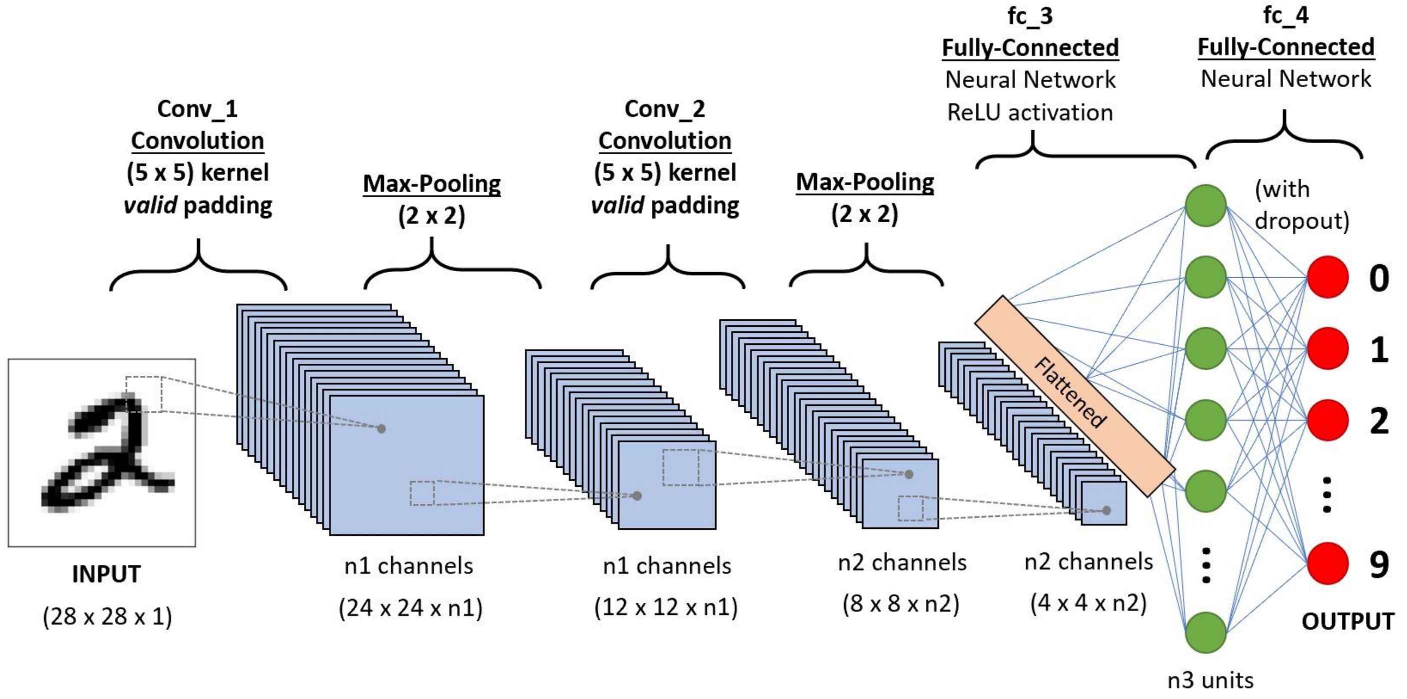

convolutional neural network

```
from keras.datasets import mnist

# Load data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Data attributes
print("train_images dimentions: ", train_images.ndim)
print("train_images shape: ", train_images.shape)
print("train_images type: ", train_images.dtype)

X_train = train_images.reshape(60000, 28, 28, 1)
X_test = test_images.reshape(10000, 28, 28, 1)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

from keras.utils import np_utils
Y_train = np_utils.to_categorical(train_labels)
Y_test = np_utils.to_categorical(test_labels)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
```

```
11493376/11490434 [==============================] - 0s 0us/step
train_images dimentions:  3
train_images shape:  (60000, 28, 28)
train_images type:  uint8
```

```
# Creating our model
from keras.models import Model
from keras import layers
import keras

myInput = layers.Input(shape=(28,28,1))
conv1 = layers.Conv2D(16, 3, activation='relu', padding='same')(myInput) #filter #window size
pool1 = layers.MaxPool2D(pool_size=2)(conv1)
conv2 = layers.Conv2D(32, 3, activation='relu', padding='same')(pool1)
pool2 = layers.MaxPool2D(pool_size=2)(conv2)

flat = layers.Flatten()(pool2)
out_layer = layers.Dense(10, activation='softmax')(flat)

myModel = Model(myInput, out_layer)

myModel.summary()
myModel.compile(optimizer=keras.optimizers.Adam(), loss=keras.losses.categorical_crossentropy
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 10)                15690
=================================================================
Total params: 20,490
Trainable params: 20,490
Non-trainable params: 0
_________________________________________________________________
```

```
network_history = myModel.fit(X_train, Y_train, batch_size=128, epochs=5, validation_split=0.
```

```
Epoch 1/5
375/375 [==============================] - 27s 70ms/step - loss: 0.8704 - accuracy: 0.74
```

```
Epoch 2/5
375/375 [==============================] - 26s 69ms/step - loss: 0.1077 - accuracy: 0.96
Epoch 3/5
375/375 [==============================] - 26s 69ms/step - loss: 0.0746 - accuracy: 0.97
Epoch 4/5
375/375 [==============================] - 26s 69ms/step - loss: 0.0570 - accuracy: 0.98
Epoch 5/5
375/375 [==============================] - 26s 69ms/step - loss: 0.0494 - accuracy: 0.98
```

```
myInput = layers.Input(shape=(28,28,1))
conv1 = layers.Conv2D(16, 3, activation='relu', padding='same', strides=2)(myInput)
conv2 = layers.Conv2D(32, 3, activation='relu', padding='same', strides=2)(conv1)
flat = layers.Flatten()(conv2)
out_layer = layers.Dense(10, activation='softmax')(flat)

myModel = Model(myInput, out_layer)

myModel.summary()
myModel.compile(optimizer=keras.optimizers.Adam(), loss=keras.losses.categorical_crossentropy
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 16)        160
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 32)          4640
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                15690
=================================================================
Total params: 20,490
Trainable params: 20,490
Non-trainable params: 0
_________________________________________________________________
```

```
network_history = myModel.fit(X_train, Y_train, batch_size=128, epochs=5, validation_split=0.
```

```
Epoch 1/5
375/375 [==============================] - 8s 20ms/step - loss: 0.9441 - accuracy: 0.754
Epoch 2/5
375/375 [==============================] - 8s 20ms/step - loss: 0.1886 - accuracy: 0.944
Epoch 3/5
375/375 [==============================] - 8s 21ms/step - loss: 0.1100 - accuracy: 0.967
Epoch 4/5
375/375 [==============================] - 8s 21ms/step - loss: 0.0815 - accuracy: 0.976
Epoch 5/5
375/375 [==============================] - 8s 21ms/step - loss: 0.0687 - accuracy: 0.979
```