**ASSINGMENT 7**

**Name: varna nemulla**

**Id: 700744920**

**Use case:**

**Use Case Description: Image Classification with CNN 1. Training the model 2. Evaluating the model Programming elements:**

**1. About CNN**

**2. Hyperparameters of CNN**

**3. Image classification with CNN**

**In class programming: 1. Follow the instruction below and then report how the performance changed.(apply all at once)**

**• Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function. • Dropout layer at 20%. • Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function. • Max Pool layer with size 2×2. • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function. • Dropout layer at 20%. • Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function. • Max Pool layer with size 2×2. • Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function. • Dropout layer at 20%. • Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function. • Max Pool layer with size 2×2. • Flatten layer. • Dropout layer at 20%. • Fully connected layer with 1024 units and a rectifier activation function. • Dropout layer at 20%. • Fully connected layer with 512 units and a rectifier activation function. • Dropout layer at 20%. • Fully connected output layer with 10 units and a Softmax activation function Did the performance change? 2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.**

**Source Code:**

**import numpy as np**

**from keras.datasets import cifar10**

**from keras.models import Sequential**

**from keras.layers import Dense, Dropout, Flatten**

**from keras.constraints import maxnorm**

**from keras.optimizers import SGD**

**from keras.layers.convolutional import Conv2D, MaxPooling2D**

**from keras.utils import np\_utils**

**np.random.seed(7)**

**(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()**

**X\_train = X\_train.astype('float32') / 255.0**

**X\_test = X\_test.astype('float32') / 255.0**

**y\_train = np\_utils.to\_categorical(y\_train)**

**y\_test = np\_utils.to\_categorical(y\_test)**

**num\_classes = y\_test.shape[1]**

**model = Sequential()**

**model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(MaxPooling2D(pool\_size=(2, 2), padding='same'))**

**model.add(Flatten())**

**model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.5))**

**model.add(Dense(num\_classes, activation='softmax'))**

**sgd = SGD(learning\_rate=0.01, momentum=0.9, decay=1e-6)**

**model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])**

**print(model.summary())**

**sgd = SGD(learning\_rate=0.01, momentum=0.9, decay=1e-6)**

**model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])**

**print(model.summary())**

**scores = model.evaluate(X\_test, y\_test, verbose=0)**

**print("Accuracy: %.2f%%" % (scores[1]\*100))**

**import numpy as np**

**from keras.datasets import cifar10**

**from keras.models import Sequential**

**from keras.layers import Dense, Dropout, Flatten**

**from keras.layers.convolutional import Conv2D, MaxPooling2D**

**from keras.constraints import maxnorm**

**from keras.utils import np\_utils**

**from keras.optimizers import SGD**

**# Fix random seed for reproducibility**

**np.random.seed(7)**

**# Load data**

**(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()**

**# Normalize inputs from 0-255 to 0.0-1.0**

**X\_train = X\_train.astype('float32') / 255.0**

**X\_test = X\_test.astype('float32') / 255.0**

**# One hot encode outputs**

**y\_train = np\_utils.to\_categorical(y\_train)**

**y\_test = np\_utils.to\_categorical(y\_test)**

**num\_classes = y\_test.shape[1]**

**# Create the model**

**model = Sequential()**

**model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(MaxPooling2D(pool\_size=(2, 2)))**

**model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(MaxPooling2D(pool\_size=(2, 2)))**

**model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))**

**model.add(MaxPooling2D(pool\_size=(2, 2)))**

**model.add(Flatten())**

**model.add(Dropout(0.2))**

**model.add(Dense(1024, activation='relu', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))**

**model.add(Dropout(0.2))**

**model.add(Dense(num\_classes, activation='softmax'))**

**# Compile model**

**epochs = 5**

**learning\_rate = 0.01**

**decay\_rate = learning\_rate / epochs**

**sgd = SGD(lr=learning\_rate, momentum=0.9, decay=decay\_rate, nesterov=False)**

**model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])**

**print(model.summary())**

**# Fit the model**

**history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=32)**

**# Evaluate the model**

**scores = model.evaluate(X\_test, y\_test, verbose=0)**

**print("Accuracy: %.2f%%" % (scores[1] \* 100))**

**# Predict the first 4 images of the test data**

**predictions = model.predict(X\_test[:4])**

**# Convert the predictions to class labels**

**predicted\_labels = numpy.argmax(predictions, axis=1)**

**# Convert the actual labels to class labels**

**actual\_labels = numpy.argmax(y\_test[:4], axis=1)**

**# Print the predicted and actual labels for the first 4 images**

**print("Predicted labels:", predicted\_labels)**

**print("Actual labels: ", actual\_labels)**

**import matplotlib.pyplot as plt**

**# Plot the training and validation loss**

**plt.plot(history.history['loss'])**

**plt.plot(history.history['val\_loss'])**

**plt.title('Model Loss')**

**plt.ylabel('Loss')**

**plt.xlabel('Epoch')**

**plt.legend(['train', 'val'], loc='upper right')**

**plt.show()**

**# Plot the training and validation accuracy**

**plt.plot(history.history['accuracy'])**

**plt.plot(history.history['val\_accuracy'])**

**plt.title('Model Accuracy')**

**plt.ylabel('Accuracy')**

**plt.xlabel('Epoch')**

**plt.legend(['train', 'val'], loc='lower right')**

**plt.show()**

**Screenshots of output:**

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, table

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

**3. Visualize Loss and Accuracy using the history object**

Graphical user interface, chart

Description automatically generated

**Description**: In this assignment I have developed this program where the Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function and the Dropout layer at 20%

Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function where the Max Pool layer with size 2×2

Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function and the Dropout layer at 20%

Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function and the Max Pool layer with size 2×2

Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function and the Dropout layer at 20%.

Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function and the Max Pool layer with size 2×2

Flatten layer Dropout layer at 20%. Fully connected layer with 1024 units and a rectifier activation function and the Dropout layer at 20%. Fully connected layer with 512 units and a rectifier activation function and the Dropout layer at 20%. Fully connected output layer with 10 units and a Softmax activation function.The performance has changed.

Predicted the first 4 images of the test data using the above model. Then, compared with the actual label for those 4 images to check whether or not the model has predicted was not correct. Visualized the Loss and Accuracy using the history object.

**Video Link:** [**https://drive.google.com/file/d/1fr6ir0Ie\_hqJj4JSa3As066ZNrVj5rzS/view?usp=sharing**](https://drive.google.com/file/d/1fr6ir0Ie_hqJj4JSa3As066ZNrVj5rzS/view?usp=sharing)

**GitHub Link:** **http://localhost:8888/notebooks/DL%20Assignment%207.ipynb**