

Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
Summer Semester	AY 2024-2024
Assignment 5.2: Build and Apply Multilayer Perceptron	
Name	Ramos, William Laurence M.
Section	CPE32S1
Date Performed:	23/06/24
Date Submitted:	23/06/24
Instructor:	Engr. Roman Richard

✓ **Part 1: Try the MLP Notebook using the CIFAR10 Keras Dataset**

```
import numpy as np

from keras.datasets import cifar10
from keras.utils import to_categorical

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

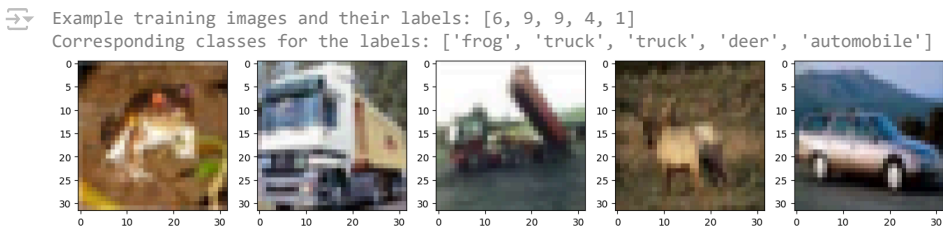
print("Shape of training data:")
print(X_train.shape)
print(y_train.shape)
print("Shape of test data:")
print(X_test.shape)
print(y_test.shape)

import matplotlib.pyplot as plt

cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
print('Example training images and their labels: ' + str([x[0] for x in y_train[0:5]]))
print('Corresponding classes for the labels: ' + str([cifar_classes[x[0]] for x in y_train[0:5]]))

f, axarr = plt.subplots(1, 5)
f.set_size_inches(16, 6)

for i in range(5):
    img = X_train[i]
    axarr[i].imshow(img)
plt.show()
```



```

y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Transform images from (32,32,3) to 3072-dimensional vectors (32*32*3)

X_train = np.reshape(X_train,(50000,3072))
X_test = np.reshape(X_test,(10000,3072))
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# Normalization of pixel values (to [0-1] range)

X_train /= 255
X_test /= 255

from keras.models import Sequential
from keras.layers import Dense, Activation
from tensorflow.keras.optimizers import SGD, schedules

model = Sequential()
model.add(Dense(256, activation='relu', input_dim=3072))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Use a learning rate schedule instead of the deprecated 'decay'
lr_schedule = schedules.ExponentialDecay(
    initial_learning_rate=0.01,
    decay_steps=10000, # Adjust this value based on your dataset and training process
    decay_rate=0.9
)
sgd = SGD(learning_rate=lr_schedule, momentum=0.9, nesterov=True)

model.compile(optimizer=sgd,loss='categorical_crossentropy',metrics=['accuracy'])

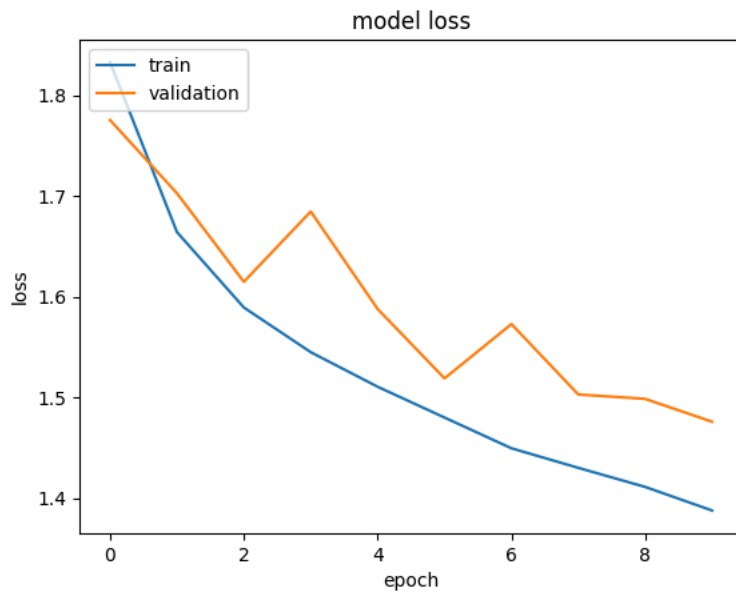
# Training the MLP

history = model.fit(X_train,y_train, epochs=10, batch_size=32, verbose=1, validation_split=0.2)

➡ Epoch 1/10
1250/1250 [=====] - 19s 14ms/step - loss: 1.8329 - accuracy: 0.3347 - val_loss: 1.7754 - val_accuracy: 0.3570
Epoch 2/10
1250/1250 [=====] - 15s 12ms/step - loss: 1.6642 - accuracy: 0.4011 - val_loss: 1.7028 - val_accuracy: 0.3786
Epoch 3/10
1250/1250 [=====] - 15s 12ms/step - loss: 1.5893 - accuracy: 0.4338 - val_loss: 1.6149 - val_accuracy: 0.4326
Epoch 4/10
1250/1250 [=====] - 18s 15ms/step - loss: 1.5450 - accuracy: 0.4475 - val_loss: 1.6847 - val_accuracy: 0.4101
Epoch 5/10
1250/1250 [=====] - 16s 13ms/step - loss: 1.5106 - accuracy: 0.4585 - val_loss: 1.5878 - val_accuracy: 0.4280
Epoch 6/10
1250/1250 [=====] - 15s 12ms/step - loss: 1.4800 - accuracy: 0.4683 - val_loss: 1.5189 - val_accuracy: 0.4630
Epoch 7/10
1250/1250 [=====] - 15s 12ms/step - loss: 1.4495 - accuracy: 0.4796 - val_loss: 1.5728 - val_accuracy: 0.4534
Epoch 8/10
1250/1250 [=====] - 18s 14ms/step - loss: 1.4301 - accuracy: 0.4839 - val_loss: 1.5029 - val_accuracy: 0.4702
Epoch 9/10
1250/1250 [=====] - 22s 17ms/step - loss: 1.4111 - accuracy: 0.4899 - val_loss: 1.4986 - val_accuracy: 0.4755
Epoch 10/10
1250/1250 [=====] - 16s 12ms/step - loss: 1.3877 - accuracy: 0.5016 - val_loss: 1.4758 - val_accuracy: 0.4766

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

```



```
# Evaluating the MLP
score = model.evaluate(X_test, y_test, batch_size=128, verbose=0)
```

```
print(model.metrics_names)
print(score)
```



```
['loss', 'accuracy']
[1.4821416139602661, 0.478300005197525]
```

✓ Part 2:

- Choose any dataset

```
import numpy as np
```

```
from keras.datasets import cifar100
from keras.utils import to_categorical
```

```
(X_train, y_train), (X_test, y_test) = cifar100.load_data()
```

✓ Explain the problem you are trying to solve

- I'm trying to do the same with cifar dataset but using cifar 100 so that it will have more data and hoping that it will be more useful in identifying different or assorted sample images with the use of cifar100 dataset.
- Create your own model

```

import matplotlib.pyplot as plt

# Define the class names for CIFAR dataset with the given structure
cifar_classes = [
    'beaver', 'dolphin', 'otter', 'seal', 'whale',
    'aquarium fish', 'flatfish', 'ray', 'shark', 'trout',
    'orchids', 'poppies', 'roses', 'sunflowers', 'tulips',
    'bottles', 'bowls', 'cans', 'cups', 'plates',
    'apples', 'mushrooms', 'oranges', 'pears', 'sweet peppers',
    'clock', 'computer keyboard', 'lamp', 'telephone', 'television',
    'bed', 'chair', 'couch', 'table', 'wardrobe',
    'bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach',
    'bear', 'leopard', 'lion', 'tiger', 'wolf',
    'bridge', 'castle', 'house', 'road', 'skyscraper',
    'cloud', 'forest', 'mountain', 'plain', 'sea',
    'camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo',
    'fox', 'porcupine', 'possum', 'raccoon', 'skunk',
    'crab', 'lobster', 'snail', 'spider', 'worm',
    'baby', 'boy', 'girl', 'man', 'woman',
    'crocodile', 'dinosaur', 'lizard', 'snake', 'turtle',
    'hamster', 'mouse', 'rabbit', 'shrew', 'squirrel',
    'maple', 'oak', 'palm', 'pine', 'willow',
    'bicycle', 'bus', 'motorcycle', 'pickup truck', 'train',
    'lawn-mower', 'rocket', 'streetcar', 'tank', 'tractor'
]

# Define the indices of animal classes
animal_indices = list(range(0, 10)) + list(range(35, 45)) + list(range(55, 70))

# Find the indices in the dataset that belong to animal classes
animal_data_indices = [i for i, label in enumerate(y_train) if label[0] in animal_indices]

# Select a subset of 5 images to display
selected_indices = animal_data_indices[:5]

# Print the labels of the training images in the selected indices
labels = [y_train[i][0] for i in selected_indices] # Access the integer values directly
print('Example training images and their labels: ' + str(labels))


# Ensure labels are within range and print the corresponding classes
class_labels = [cifar_classes[label] for label in labels]
print('Corresponding classes for the labels: ' + str(class_labels))

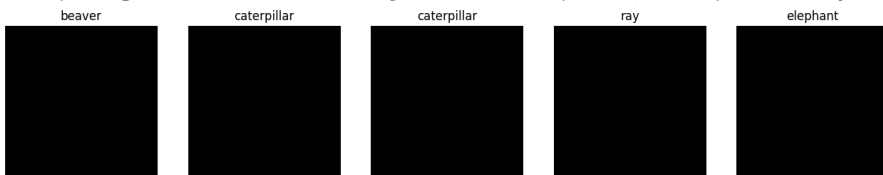
# Create a figure with 1 row and 5 columns of subplots
f, axarr = plt.subplots(1, 5)
f.set_size_inches(16, 6)
# Reshape the image data to its original dimensions (32x32x3)
for i, idx in enumerate(selected_indices):
    img = X_train[idx].reshape(32, 32, 3) # Reshape to 32x32 pixels with 3 color channels
    axarr[i].imshow(img)
    axarr[i].set_title(class_labels[i]) # Set the title as the class name

# Remove axes ticks for clarity
for ax in axarr:
    ax.axis('off')

# Display the figure with the images
plt.show()

```

 Example training images and their labels: [0, 38, 38, 7, 58]
 Corresponding classes for the labels: ['beaver', 'caterpillar', 'caterpillar', 'ray', 'e



- Evaluate the accuracy of your model

```
import numpy as np
from keras.utils import to_categorical

# Assuming you have loaded X_train, y_train, X_test, y_test from CIFAR dataset or elsewhere

# Example: if your labels range from 0 to 59 (total 60 classes)
num_classes = 60

# Ensure y_train and y_test are integers (convert if necessary)
y_train = np.array(y_train, dtype=np.int32)
y_test = np.array(y_test, dtype=np.int32)

# Check the range of labels before one-hot encoding
print("Minimum label value in y_train:", np.min(y_train))
print("Maximum label value in y_train:", np.max(y_train))

# Adjust labels to be in the range [0, num_classes - 1] if necessary
# For example, if your original labels were in the range [1, 60], subtract 1:
y_train = y_train - 1 # Adjust labels to start from 0
y_test = y_test - 1 # Adjust labels to start from 0

# Check unique values in y_train and y_test after adjustment
unique_train_labels = np.unique(y_train)
unique_test_labels = np.unique(y_test)
print('Unique labels in y_train:', unique_train_labels)
print('Unique labels in y_test:', unique_test_labels)

# Reshape images from (32, 32, 3) to 3072-dimensional vectors (32*32*3)
X_train = np.reshape(X_train, (X_train.shape[0], 32*32*3))
X_test = np.reshape(X_test, (X_test.shape[0], 32*32*3))

# Convert to float32 and normalize pixel values to [0, 1] range
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255

# One-hot encode the labels
y_train = to_categorical(y_train, num_classes=num_classes)
y_test = to_categorical(y_test, num_classes=num_classes)

# Print shape of the arrays to verify
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
print('X_test shape:', X_test.shape)
print('y_test shape:', y_test.shape)
```

```

↩ Minimum label value in y_train: -2
Maximum label value in y_train: 97
Unique labels in y_train: [-3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 1
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92
93 94 95 96]
Unique labels in y_test: [-3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92
93 94 95 96]

```

IndexError Traceback (most recent call last)
[ipython-input-21-eae5091fa9bf](#) in <cell line: 37>()

```

35
36 # One-hot encode the labels
--> 37 y_train = to_categorical(y_train, num_classes=num_classes)
38 y_test = to_categorical(y_test, num_classes=num_classes)
39

```

```

/usr/local/lib/python3.10/dist-packages/keras/src/utils/np_utils.py in
to_categorical(y, num_classes, dtype)
72     n = y.shape[0]
73     categorical = np.zeros((n, num_classes), dtype=dtype)
--> 74     categorical[np.arange(n), y] = 1
75     output_shape = input_shape + (num_classes,)
76     categorical = np.reshape(categorical, output_shape)

```

IndexError: index 83 is out of bounds for axis 1 with size 60

Next steps: [Explain error](#)

```

from keras.models import Sequential
from keras.layers import Dense, Activation
from tensorflow.keras.optimizers import SGD, schedules

model = Sequential()
model.add(Dense(256, activation='relu', input_dim=3072))
model.add(Dense(256, activation='relu'))
model.add(Dense(60, activation='softmax')) # Change the number of output units to 60

# Use a learning rate schedule instead of the deprecated 'decay'
lr_schedule = schedules.ExponentialDecay(
    initial_learning_rate=0.01,
    decay_steps=10000, # Adjust this value based on your dataset and training process
    decay_rate=0.9
)
sgd = SGD(learning_rate=lr_schedule, momentum=0.9, nesterov=True)

model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])

# Training the MLP

history = model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1, validation_split=0.2)

```

↩ Epoch 1/10

ValueError Traceback (most recent call last)

[<ipython-input-113-418adf6786fb>](#) in <cell line: 3>()

1 # Training the MLP

2

----> 3 history = model.fit(X_train,y_train, epochs=10, batch_size=32, verbose=1, validation_split=0.2)

↕ 1 frames

[/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py](#) in tf__train_function(iterator)

13 try:

14 do_return = True