ICP 5

KAVYA REDDY NAGULAPALLY
700759486

Github link: https://github.com/Kavyareddy03/ICP-5

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
import numpy as np
from keras.datasets import cifar10
from keras.src.utils import to_categorical
import matplotlib.pyplot as plt

seed = 7
np.random.seed(seed)
# 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')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
# one hot encode outputs
# use to_categorical directly instead of np_utils.to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Initialize the model
model = Sequential()

# 1st Convolutional Layer (32 feature maps, 3x3 kernel, ReLU activation)
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))  # Adjust input shape for your data
```

```
[1]
# Max Pooling Layer (2x2 pool size)
model.add(MaxPooling2D(pool_size=(2, 2)))

# 2nd Convolutional Layer (64 feature maps, 3x3 kernel, ReLU activation)
model.add(Conv2D(64, (3, 3), activation='relu'))

# Max Pooling Layer (2x2 pool size)
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten layer
model.add(Flatten())

# Fully Connected Layer (512 units, ReLU activation)
model.add(Dense(512, activation='relu'))

# Dropout (50%)
model.add(Dropout(0.5))

# Output Layer (10 units, Softmax activation)
model.add(Dense(10, activation='softmax'))

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

# Summary of the model
model.summary()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argume
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 512)	1,180,160
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5,130

```
Total params: 1,204,682 (4.60 MB)
Trainable params: 1,204,682 (4.60 MB)
Non-trainable params: 0 (0.00 B)
```

Explanation:

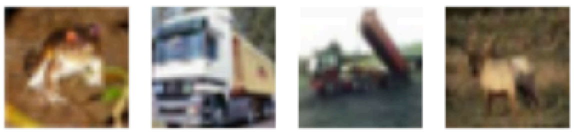
1. Data Preparation: It loads the CIFAR-10 dataset, normalizes the pixel values from 0-255 to 0-1, and applies one-hot encoding to the labels.

2. Model Definition: A Sequential model is created with:
   - Convolutional Layers: Two layers that apply convolution operations to extract features from the images, each followed by a ReLU activation function.
   - Max Pooling Layers: Two layers that downsample the feature maps, reducing spatial dimensions while retaining important features.
   - Flatten Layer: Converts the 2D feature maps into a 1D vector.
   - Dense Layer: A fully connected layer with 512 units and ReLU activation for learning complex patterns.
   - Dropout Layer: Regularizes the model by randomly dropping 50% of the neurons during training to prevent overfitting.
   - Output Layer: A dense layer with 10 units and softmax activation for multi-class classification.

3. Model Compilation: The model is compiled with the Adam optimizer, categorical crossentropy loss, and accuracy as the evaluation metric.

4. Model Summary: A summary of the model architecture is printed to show the layers and parameters.

```
import matplotlib.pyplot as plt
plt.figure(figsize = (4,4))
for i in range(4):
  plt.subplot(3,5,1+i)
  plt.axis('off')
  plt.imshow(X_train[i], cmap = 'gray')
```



Explanation:

1. Figure Setup: It creates a 4x4 inch figure.
2. Subplot Loop: It iterates over the first four images to create subplots in a 3-row by 5-column grid.
3. Axis Management: The axes are turned off for each subplot to focus solely on the images.
4. Image Display: Each image is displayed using a grayscale colormap, though the images are originally colored.

```
[3] import numpy as np
    # Predict the first 4 images of the test data
    y_predictions = model.predict(X_test[:4])

    #y_predictions.reshape(-1,)
    y_predictions= np.argmax(y_predictions, axis=1)
    # Convert the actual labels to class labels
    actual_labels = np.argmax(y_test[:4], axis=1)

    print("Predicted labels:", y_predictions)
    print("Actual labels:   ", actual_labels)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 320ms/step
Predicted labels: [4 4 4 4]
Actual labels:    [3 8 8 0]
```

Explanation:

Prediction: It uses the model to predict the labels for the first four images in `X_test`.

Label Conversion: The predicted probabilities are converted to class labels by using `argmax`, which identifies the index of the highest probability for each prediction.

Actual Labels: The true labels of the first four images are also converted to class labels using `argmax`.

Output: Finally, it prints the predicted labels and the actual labels for comparison.

```python
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
L = 2
W = 2
fig, axes = plt.subplots(L, W, figsize = (6,6))
axes = axes.ravel() #

for i in np.arange(0, 4):
    axes[i].imshow(X_test[i])
    axes[i].set_title("Predicted = {}\n Actual  = {}".format(classes[y_predictions[i]], classes[actual_labels[i]]))
    axes[i].axis('off')

plt.subplots_adjust(wspace=1)
```

Explanation:

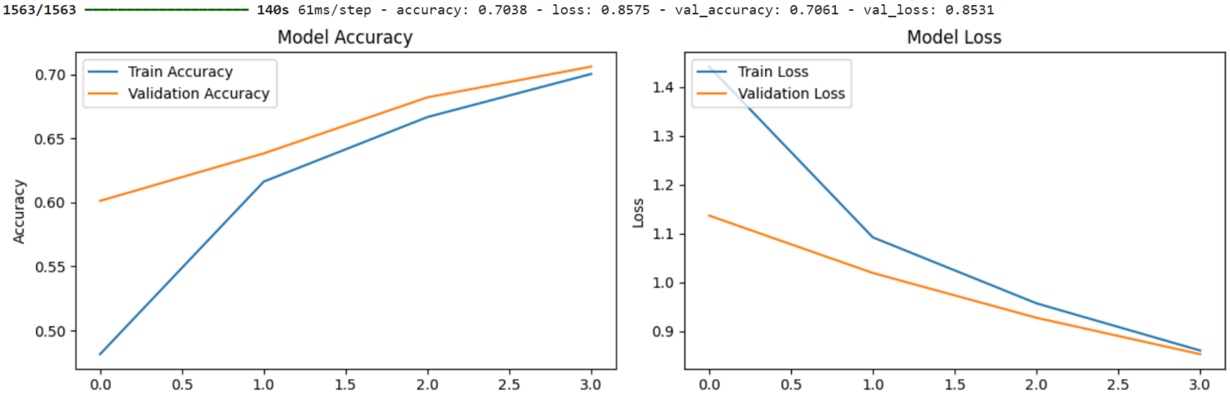
Class Labels: A list of class names for the CIFAR-10 dataset is defined.
Figure Setup: A 2x2 grid of subplots is created for displaying images.
Image Loop: It iterates over the first four images:

- Each image is displayed in its corresponding subplot.
- The title for each subplot shows the predicted and actual class labels.
- The axis is turned off for a cleaner look.

Spacing Adjustment: It adjusts the space between subplots for better visualization

```python
import matplotlib.pyplot as plt

# Assuming 'history' is the object returned from model.fit()
history = model.fit(X_train, y_train, epochs=4, validation_data=(X_test, y_test))

# Plotting training & validation accuracy values
plt.figure(figsize=(12, 4))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='upper left')

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper left')
```

```
Epoch 1/4
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 108s 66ms/step - accuracy: 0.3853 - loss: 1.6721 - val_accuracy: 0.6012 - val_loss: 1.1367
Epoch 2/4
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 98s 63ms/step - accuracy: 0.6023 - loss: 1.1224 - val_accuracy: 0.6383 - val_loss: 1.0193
Epoch 3/4
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 98s 63ms/step - accuracy: 0.6629 - loss: 0.9641 - val_accuracy: 0.6821 - val_loss: 0.9276
Epoch 4/4
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 140s 61ms/step - accuracy: 0.7038 - loss: 0.8575 - val_accuracy: 0.7061 - val_loss: 0.8531
```



Explanation:

Model Training: The model is trained using the training data for four epochs, with validation data also provided to monitor performance.
Figure Setup: A figure is created with a specified size to accommodate two subplots side by side.
Accuracy Plot:

- The first subplot shows the training and validation accuracy over the epochs.
- Lines are plotted for both metrics, and the axes are labeled accordingly.
- A legend is included to differentiate between training and validation accuracy.

Loss Plot:

- The second subplot displays the training and validation loss over the epochs.
- Similar to the accuracy plot, lines are added, and the axes are labeled.

Layout Adjustment: The layout is adjusted for better spacing, and the plots are displayed