Week 5 Learning Activities

Load and split the dataset into train and test

```
from tensorflow.keras.datasets import cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

Label the classes and observe the data

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'tru

from matplotlib import pyplot as plt
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

Create the model with convolutional and max pooling layers

```python
from tensorflow.keras import Model, Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.activations import relu

model: Model = Sequential()
model.add(Conv2D(32, (3, 3), activation=relu, input_shape=(32, 32, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation=relu))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation=relu))

model.summary()
```

```
/home/hailq/Documents/university-academic-archive/my-code-demo/python/.venv/lib/python3.12/site-p
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36,928

Total params: 56,320 (220.00 KB)

Trainable params: 56,320 (220.00 KB)

Add fully connected layers and a flatten layer to the model

```
model.add(Flatten())
model.add(Dense(64, activation=relu))
model.add(Dense(10))

model.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36,928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65,600
dense_1 (Dense)	(None, 10)	650

```
Total params: 122,570 (478.79 KB)

Trainable params: 122,570 (478.79 KB)
```

Fit the data to the model

```
from tensorflow.keras.losses import SparseCategoricalCrossentropy
model.compile(
    optimizer='adam',
    loss=SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

from tensorflow.keras.callbacks import History
history: History = model.fit(train_images, train_labels, epochs=10, validation_data=(test_imag
```

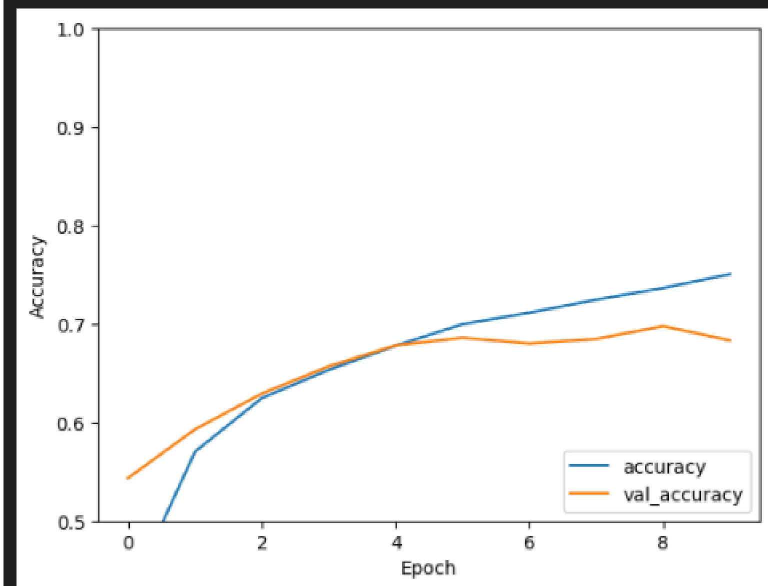
```
Epoch 1/10
2024-10-03 12:22:40.006073: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
   9/1563 ━━━━━━━━━━━━━━━━━━━━ 37s 24ms/step - accuracy: 0.1314 - loss: 2.3147
2024-10-03 12:22:42.451588: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
2024-10-03 12:22:42.451775: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
2024-10-03 12:22:42.474894: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
2024-10-03 12:22:42.474979: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 33s 20ms/step - accuracy: 0.3173 - loss: 1.8454 - val_accuracy: 0
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 29s 19ms/step - accuracy: 0.5583 - loss: 1.2370 - val_accuracy: 0
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 30s 19ms/step - accuracy: 0.6158 - loss: 1.0841 - val_accuracy: 0
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 31s 20ms/step - accuracy: 0.6523 - loss: 0.9910 - val_accuracy: 0
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.6763 - loss: 0.9282 - val_accuracy: 0
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.7016 - loss: 0.8545 - val_accuracy: 0
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.7169 - loss: 0.8071 - val_accuracy: 0
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 28s 18ms/step - accuracy: 0.7288 - loss: 0.7767 - val_accuracy: 0
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 25s 16ms/step - accuracy: 0.7384 - loss: 0.7346 - val_accuracy: 0
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 24s 15ms/step - accuracy: 0.7553 - loss: 0.7015 - val_accuracy: 0
```

Observe the training process

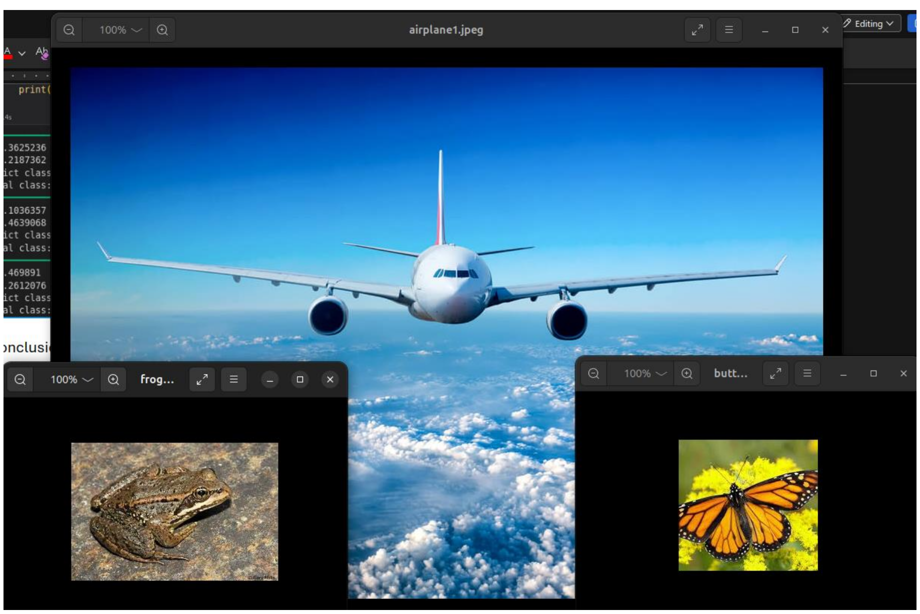
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_accuracy = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 2s - 6ms/step - accuracy: 0.6833 - loss: 0.9256
```



Test the model with real data



```python
from numpy import expand_dims, argmax
from PIL import Image
from tensorflow.keras.preprocessing import image
from os import listdir

image_dir = '/home/hailq/Documents/university-academic-archive/my-code-demo/python/datasets/i
for filename in listdir(image_dir):
    img = Image.open(image_dir + filename)
    img = img.resize((32, 32))
    img = image.img_to_array(img)
    img = expand_dims(img, axis=0)
    img = img / 255.0

    pred = model.predict(img)
    print(pred)
    print('Predict class: ', class_names[argmax(pred)])
    print('Actual class: ', filename.split('.')[0][:-1])
```

✓ 0.2s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
[[ 3.1290996  -4.784229    1.4046986  -2.8453085  -0.58353823 -2.838536
  -2.299458   -3.7307475   2.4276028  -2.38103    ]]
Predict class:  airplane
Actual class:  airplane
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
[[-2.2419212 -2.2770765 -8.08184   -1.0602847 -3.6699076 -2.6470006
   1.2513139 -2.5129309 -8.3191595  2.8426383]]
Predict class:  truck
Actual class:  butterfly
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
[[ -6.092617    -4.329122     0.2626506    0.20874931   -0.40738147
   -2.9513164   11.87858    -11.730498    -4.877231    -7.152974   ]]
Predict class:  frog
Actual class:  frog
```

Apart from the convolutional and max pooling layers, we might want to try adding several available components provided in tensorflow.keras.layers such as BatchNormalization and Dropout

```python
model: Model = Sequential([
    # First Convolutional Block
    Conv2D(32, (3, 3), activation=relu, input_shape=(32, 32, 3)),
    BatchNormalization(),  # Apply BatchNorm after activation
    MaxPooling2D((2, 2)),
    Dropout(0.25),  # Apply Dropout with a rate of 0.25

    # Second Convolutional Block
    Conv2D(64, (3, 3), activation=relu),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    # Third Convolutional Block
    Conv2D(64, (3, 3), activation=relu),
    BatchNormalization(),
    Dropout(0.25),

    # Flatten Layer
    Flatten(),

    # First Dense Block
    Dense(64, activation=relu),
    BatchNormalization(),
    Dropout(0.5),  # Apply Dropout with a rate of 0.5

    # Second Dense Block
    Dense(32, activation=relu),
    BatchNormalization(),
    Dropout(0.5),

    # Output Layer (for classification into 10 classes, assuming no activation function here)
    Dense(10)])

model.summary()
```

We notice that now there are non-trainable parameters in the model, this is due to the present of the BatchNormalization. The training time also increases slightly significantly.

```
Total params: 125,354 (489.66 KB)

Trainable params: 124,842 (487.66 KB)

Non-trainable params: 512 (2.00 KB)
```

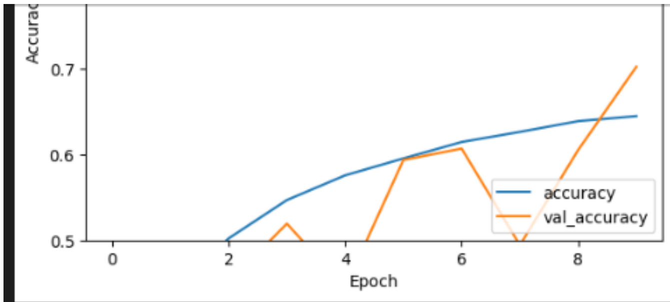
```python
from tensorflow.keras.losses import SparseCategoricalCrossentropy
model.compile(
    optimizer='adam',
    loss=SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

from tensorflow.keras.callbacks import History
history: History = model.fit(train_images, train_labels, epochs=10, validation_data=(test_ima
```

2m 24.7s

```
2024-10-03 15:15:33.699375: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocat
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 73s 43ms/step - accuracy: 0.2365 - loss: 2.2950 - val_accuracy: 0
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.4120 - loss: 1.6108
```

Eventually we observe the training process data to see whether the newly added layers have improved the training process. After that we try on the 3 images downloaded from online source.




```python
from numpy import expand_dims, argmax
from PIL import Image
from tensorflow.keras.preprocessing import image
from os import listdir

image_dir = '/home/hailq/Documents/university-academic-archive/my-code-demo/python/datasets/i
for filename in listdir(image_dir):
    img = Image.open(image_dir + filename)
    img = img.resize((32, 32))
    img = image.img_to_array(img)
    img = expand_dims(img, axis=0)
    img = img / 255.0

    pred = model.predict(img)
    print(pred)
    print('Predict class: ', class_names[argmax(pred)])
    print('Actual class: ', filename.split('.')[0][:-1])
```

✓ 0.4s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step
[[ 3.3625236  -1.2845306   0.23650116 -0.40523094 -0.38059118 -1.5617714
  -2.2187362  -1.2790147   1.9964298   0.262414  ]]
Predict class:  airplane
Actual class:  airplane
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
[[-1.1036357   0.42018843  0.42557514 -0.00677305  0.01346329 -1.2132112
   2.4639068  -1.8829615  -1.557062   -0.0977563 ]]
Predict class:  frog
Actual class:  butterfly
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
[[-3.469891   -3.617744    1.0755323   0.6849691   0.16020805 -2.0524096
   6.2612076  -4.2662783  -2.6186876  -3.9510002 ]]
Predict class:  frog
Actual class:  frog
```

In conclusion, with the modifications to the model, the training accuracy is lower, likely due to the introduction of dropout in every layer, which adds regularization and prevents the model from overfitting by randomly disabling neurons during training. However, the validation accuracy shows a consistently stable improvement throughout the training phase, indicating better generalization to unseen data and reducing the risk of overfitting.