המכללה
האקדמית להנדסה
בראודה
בכרמיאל

המחלקה להנדסת חשמל ואלקטרוניקה

(31245) מערכות לומדות ולמידה עמוקה

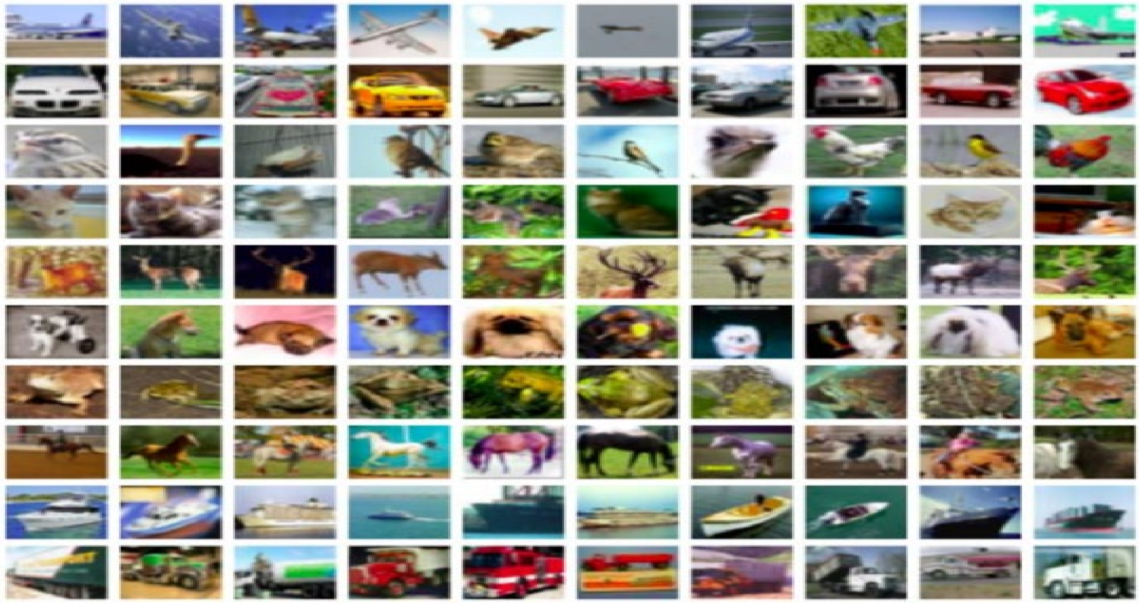
Lab 7 report

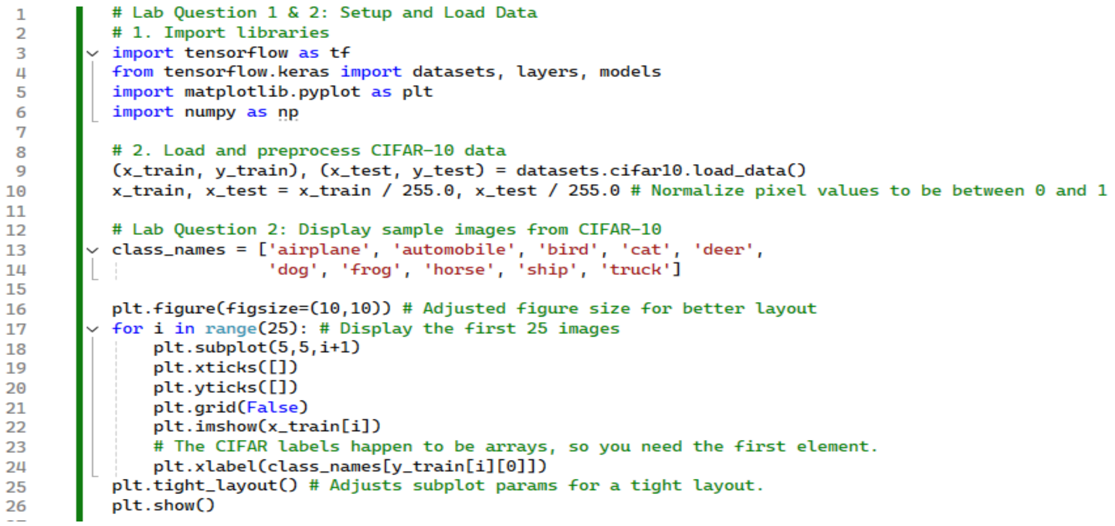
פרנסיס עבוד

Prepared for: Dr. Amer Adler

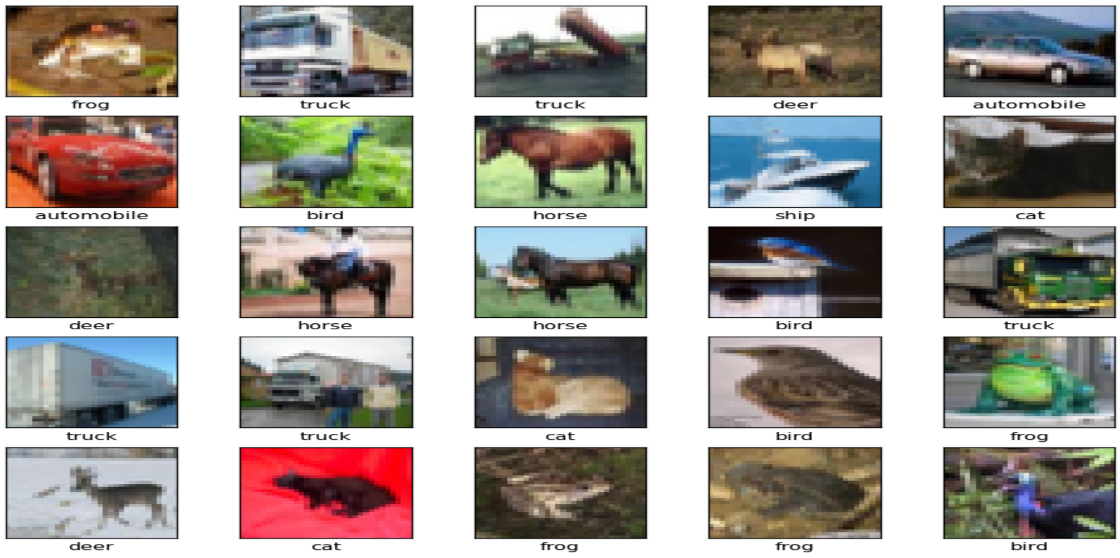
Date: 24/05/2025

In this lab we will implement a CNN for classifying 10 objects classes from CIFAR10 dataset, each image in the datasets is 32 X 32 pixels in RGB:





```python
# Lab Question 1 & 2: Setup and Load Data
# 1. Import libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np

# 2. Load and preprocess CIFAR-10 data
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0 # Normalize pixel values to be between 0 and 1

# Lab Question 2: Display sample images from CIFAR-10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10)) # Adjusted figure size for better layout
for i in range(25): # Display the first 25 images
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    # The CIFAR labels happen to be arrays, so you need the first element.
    plt.xlabel(class_names[y_train[i][0]])
plt.tight_layout() # Adjusts subplot params for a tight layout.
plt.show()
```

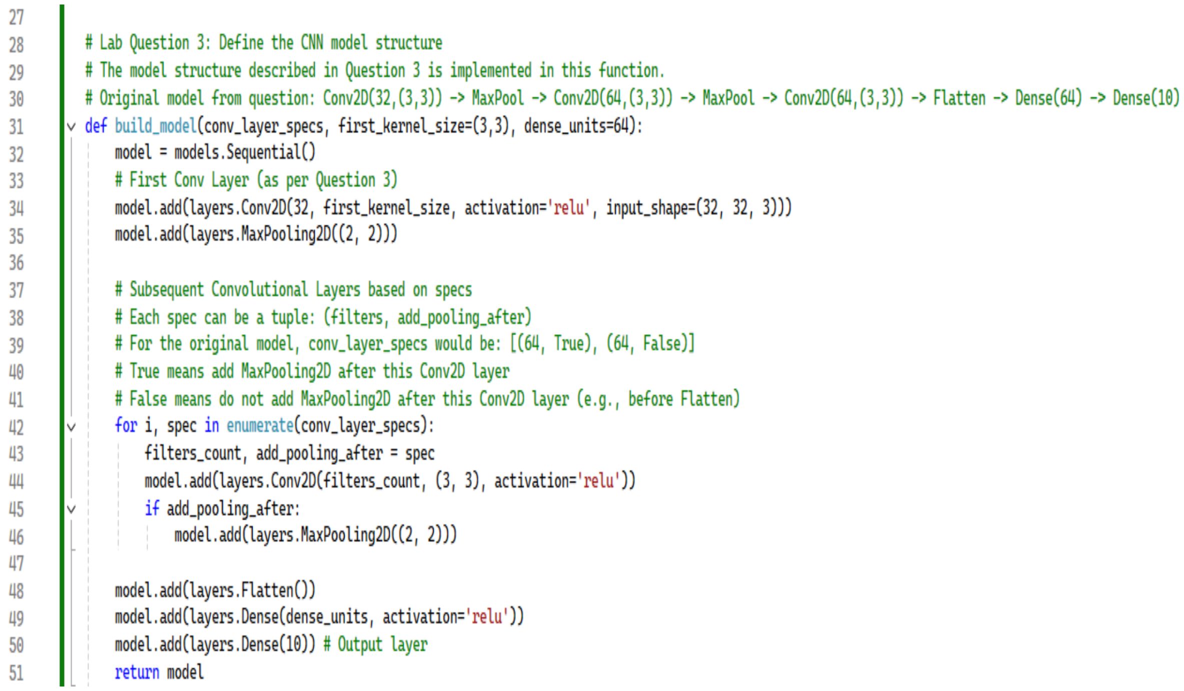


Ex.3:

The network model is given by:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10)
```

Code:



```python

# Lab Question 3: Define the CNN model structure
# The model structure described in Question 3 is implemented in this function.
# Original model from question: Conv2D(32,(3,3)) -> MaxPool -> Conv2D(64,(3,3)) -> MaxPool -> Conv2D(64,(3,3)) -> Flatten -> Dense(64) -> Dense(10)
def build_model(conv_layer_specs, first_kernel_size=(3,3), dense_units=64):
    model = models.Sequential()
    # First Conv Layer (as per Question 3)
    model.add(layers.Conv2D(32, first_kernel_size, activation='relu', input_shape=(32, 32, 3)))
    model.add(layers.MaxPooling2D((2, 2)))

    # Subsequent Convolutional Layers based on specs
    # Each spec can be a tuple: (filters, add_pooling_after)
    # For the original model, conv_layer_specs would be: [(64, True), (64, False)]
    # True means add MaxPooling2D after this Conv2D layer
    # False means do not add MaxPooling2D after this Conv2D layer (e.g., before Flatten)
    for i, spec in enumerate(conv_layer_specs):
        filters_count, add_pooling_after = spec
        model.add(layers.Conv2D(filters_count, (3, 3), activation='relu'))
        if add_pooling_after:
            model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(dense_units, activation='relu'))
    model.add(layers.Dense(10)) # Output layer
    return model
```

Ex.4:

Train the given network, using stochastic gradient descent with 20 epochs:

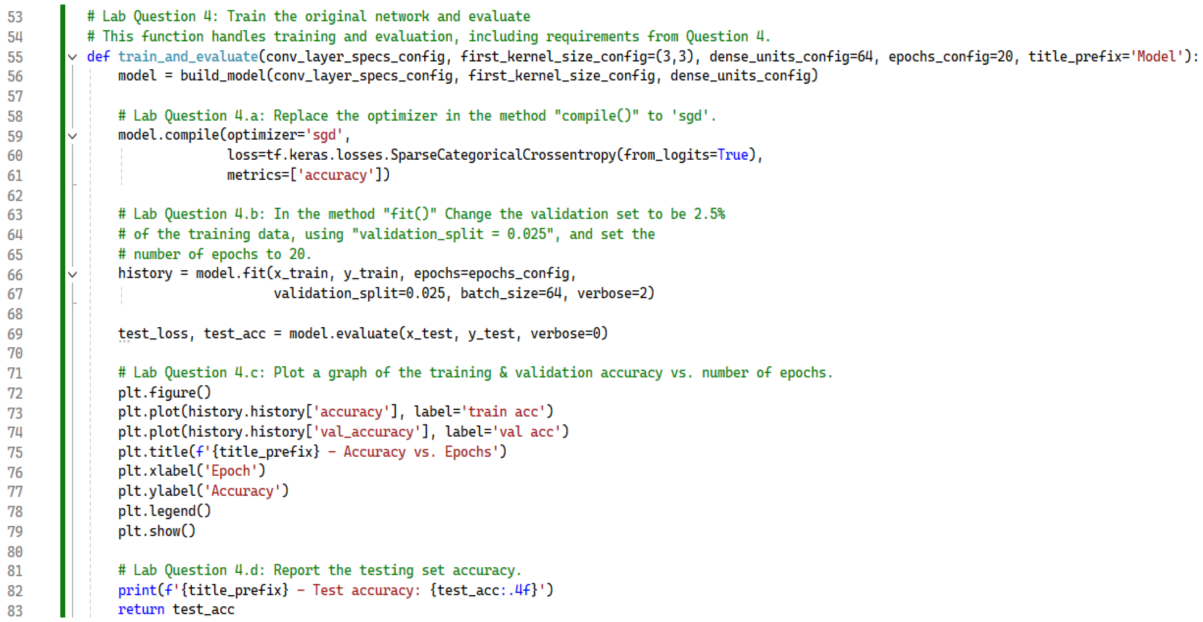
a. Replace the optimizer in the method “compile()” to ‘sgd’.

b. In the method “fit()” Change the validation set to be 2.5% of the training data, using “validation_split = 0.025”, and set the number of epochs to 20.

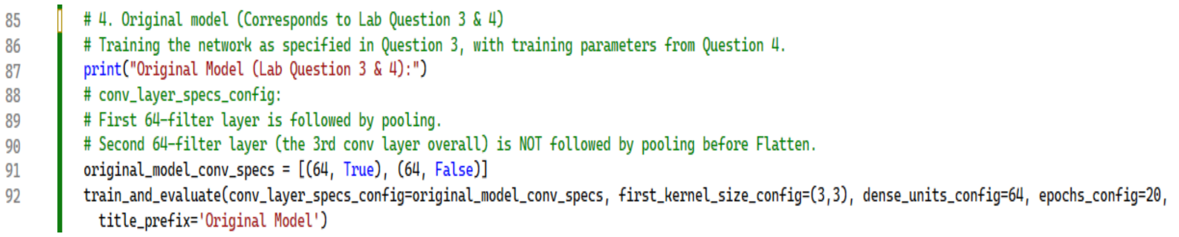
c. Plot a graph of the training & validation accuracy vs. number of epochs.

d. Report the testing set accuracy.

Code:



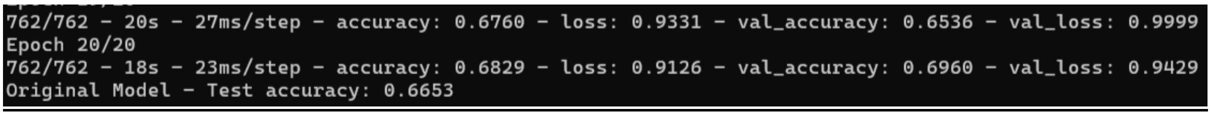
```python
# Lab Question 4: Train the original network and evaluate
# This function handles training and evaluation, including requirements from Question 4.
def train_and_evaluate(conv_layer_specs_config, first_kernel_size_config=(3,3), dense_units_config=64, epochs_config=20, title_prefix='Model'):
    model = build_model(conv_layer_specs_config, first_kernel_size_config, dense_units_config)

    # Lab Question 4.a: Replace the optimizer in the method "compile()" to 'sgd'.
    model.compile(optimizer='sgd',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    # Lab Question 4.b: In the method "fit()" Change the validation set to be 2.5%
    # of the training data, using "validation_split = 0.025", and set the
    # number of epochs to 20.
    history = model.fit(x_train, y_train, epochs=epochs_config,
                        validation_split=0.025, batch_size=64, verbose=2)

    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)

    # Lab Question 4.c: Plot a graph of the training & validation accuracy vs. number of epochs.
    plt.figure()
    plt.plot(history.history['accuracy'], label='train acc')
    plt.plot(history.history['val_accuracy'], label='val acc')
    plt.title(f'{title_prefix} - Accuracy vs. Epochs')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    # Lab Question 4.d: Report the testing set accuracy.
    print(f'{title_prefix} - Test accuracy: {test_acc:.4f}')
    return test_acc
```



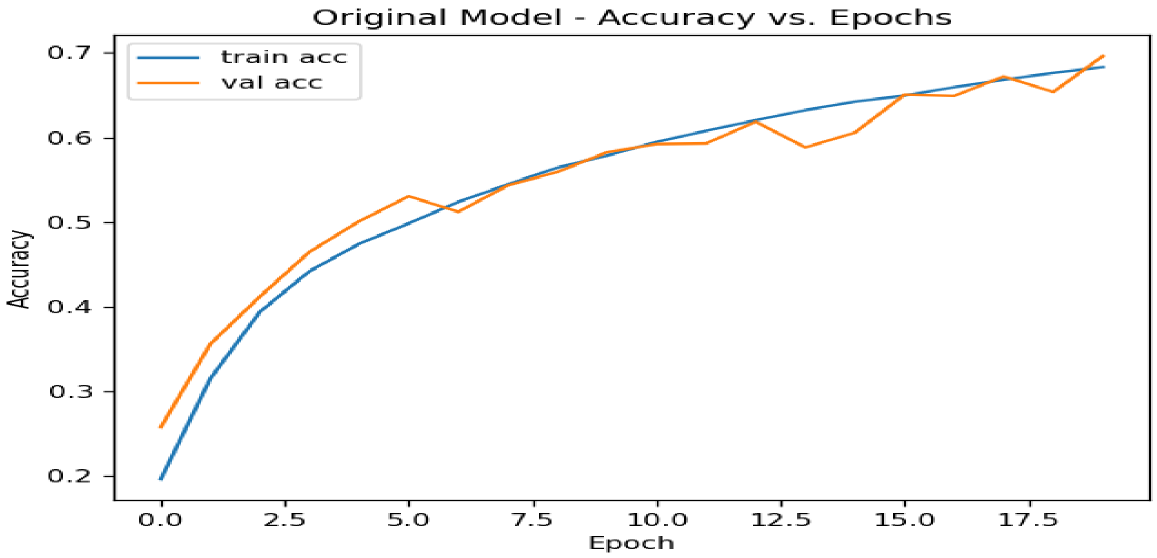
```python
# 4. Original model (Corresponds to Lab Question 3 & 4)
# Training the network as specified in Question 3, with training parameters from Question 4.
print("Original Model (Lab Question 3 & 4):")
# conv_layer_specs_config:
# First 64-filter layer is followed by pooling.
# Second 64-filter layer (the 3rd conv layer overall) is NOT followed by pooling before Flatten.
original_model_conv_specs = [(64, True), (64, False)]
train_and_evaluate(conv_layer_specs_config=original_model_conv_specs, first_kernel_size_config=(3,3), dense_units_config=64, epochs_config=20,
  title_prefix='Original Model')
```

Output:

Test accuracy= 0.6653



```
762/762 - 20s - 27ms/step - accuracy: 0.6760 - loss: 0.9331 - val_accuracy: 0.6536 - val_loss: 0.9999
Epoch 20/20
762/762 - 18s - 23ms/step - accuracy: 0.6829 - loss: 0.9126 - val_accuracy: 0.6960 - val_loss: 0.9429
Original Model - Test accuracy: 0.6653
```

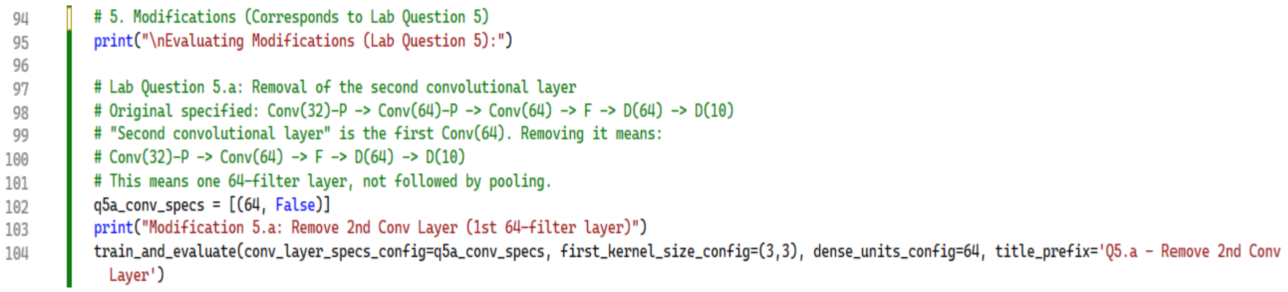


Ex.5:

Evaluate the impact on CIFAR-10 classification accuracy (measured on the testing set) of the following modifications to the network (you must re-define the CNN for each modification, otherwise, learned weights are kept). For each modification Plot a graph of the training & validation accuracy vs. number of epochs.

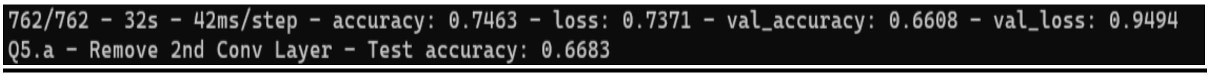
a. Removal of the second convolutional layer
b. Removal of the third convolutional layer
c. Removal of the second and third convolutional layer
d. Increase and decrease the kernel size of the first CNN layer.
e. Increase and decrease the kernel size of the first convolutional layer.
f. Increase and decrease of the number of perceptrons in the Dense layer (currently 64).

A.Code:



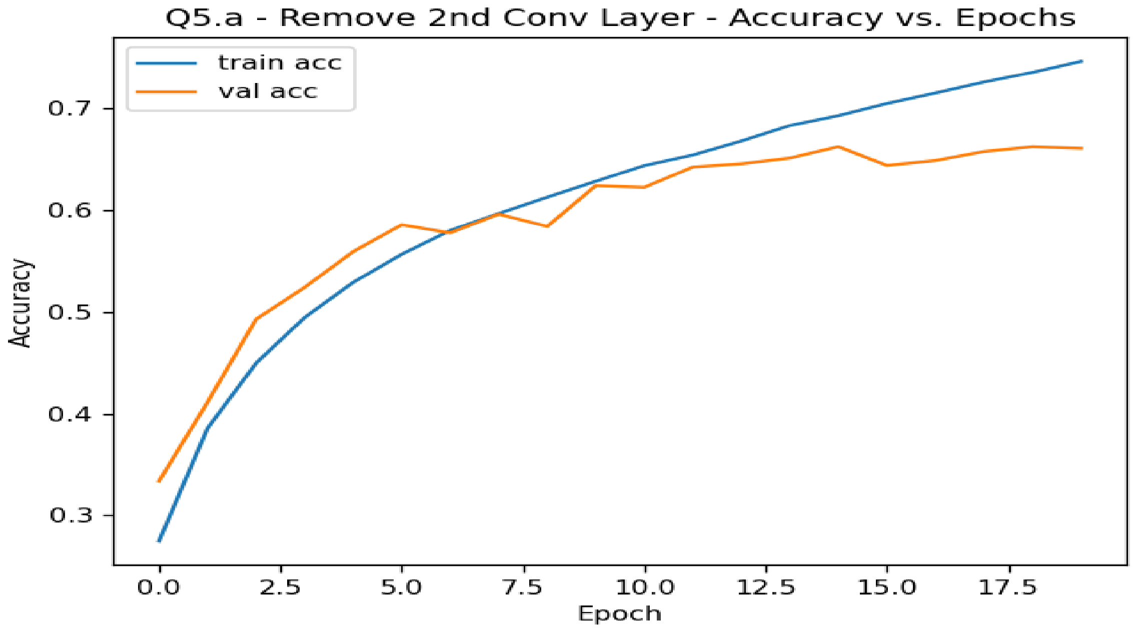
```python
# 5. Modifications (Corresponds to Lab Question 5)
print("\nEvaluating Modifications (Lab Question 5):")

# Lab Question 5.a: Removal of the second convolutional layer
# Original specified: Conv(32)-P -> Conv(64)-P -> Conv(64) -> F -> D(64) -> D(10)
# "Second convolutional layer" is the first Conv(64). Removing it means:
# Conv(32)-P -> Conv(64) -> F -> D(64) -> D(10)
# This means one 64-filter layer, not followed by pooling.
q5a_conv_specs = [(64, False)]
print("Modification 5.a: Remove 2nd Conv Layer (1st 64-filter layer)")
train_and_evaluate(conv_layer_specs_config=q5a_conv_specs, first_kernel_size_config=(3,3), dense_units_config=64, title_prefix='Q5.a - Remove 2nd Conv Layer')
```

A.Output:

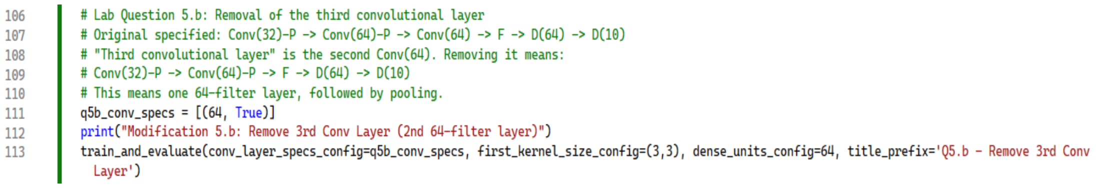


```
762/762 - 32s - 42ms/step - accuracy: 0.7463 - loss: 0.7371 - val_accuracy: 0.6608 - val_loss: 0.9494
Q5.a - Remove 2nd Conv Layer - Test accuracy: 0.6683
```

Test accuracy = 0.6683

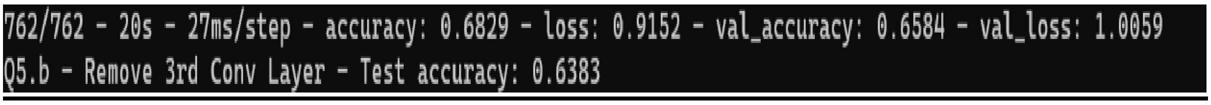


B.Code:

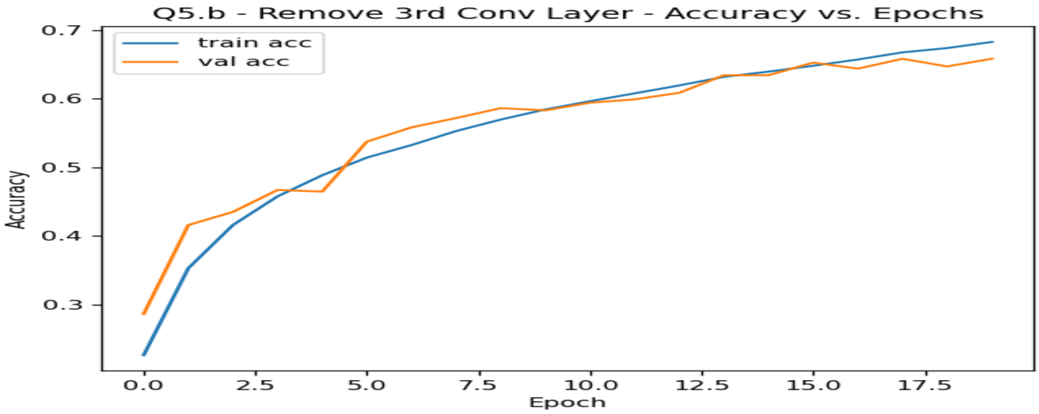


```
# Lab Question 5.b: Removal of the third convolutional layer
# Original specified: Conv(32)-P -> Conv(64)-P -> Conv(64) -> F -> D(64) -> D(10)
# "Third convolutional layer" is the second Conv(64). Removing it means:
# Conv(32)-P -> Conv(64)-P -> F -> D(64) -> D(10)
# This means one 64-filter layer, followed by pooling.
q5b_conv_specs = [(64, True)]
print("Modification 5.b: Remove 3rd Conv Layer (2nd 64-filter layer)")
train_and_evaluate(conv_layer_specs_config=q5b_conv_specs, first_kernel_size_config=(3,3), dense_units_config=64, title_prefix='Q5.b - Remove 3rd Conv
  Layer')
```

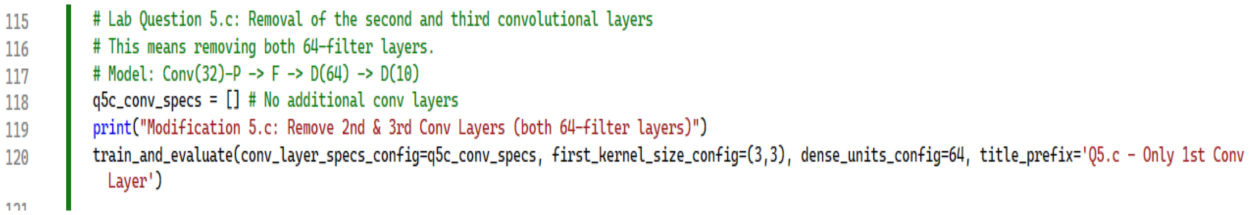
B.Output:



```
762/762 - 20s - 27ms/step - accuracy: 0.6829 - loss: 0.9152 - val_accuracy: 0.6584 - val_loss: 1.0059
Q5.b - Remove 3rd Conv Layer - Test accuracy: 0.6383
```

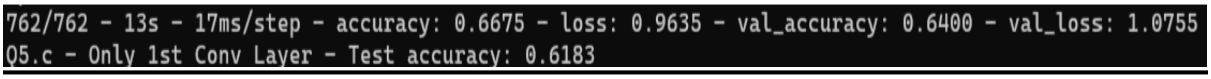


C.Code:

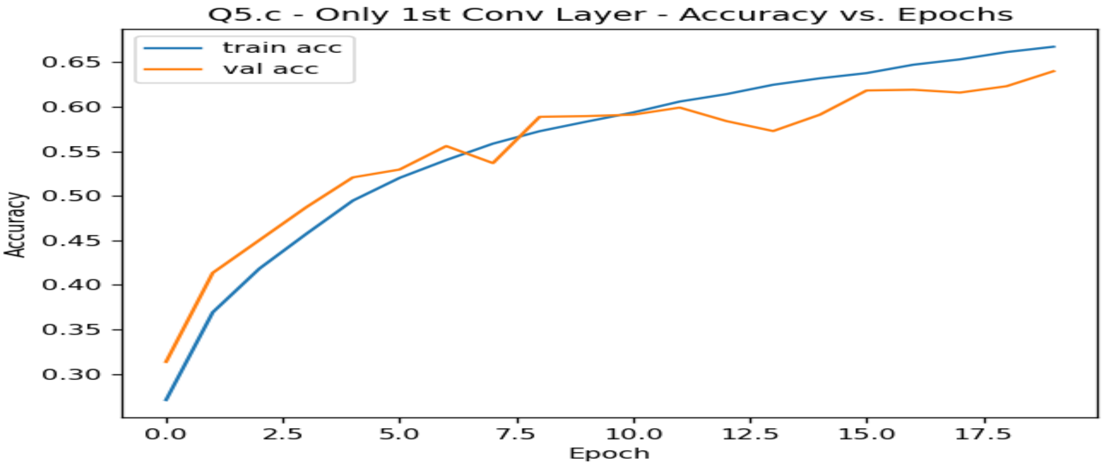


```
# Lab Question 5.c: Removal of the second and third convolutional layers
# This means removing both 64-filter layers.
# Model: Conv(32)-P -> F -> D(64) -> D(10)
q5c_conv_specs = [] # No additional conv layers
print("Modification 5.c: Remove 2nd & 3rd Conv Layers (both 64-filter layers)")
train_and_evaluate(conv_layer_specs_config=q5c_conv_specs, first_kernel_size_config=(3,3), dense_units_config=64, title_prefix='Q5.c - Only 1st Conv
  Layer')
```

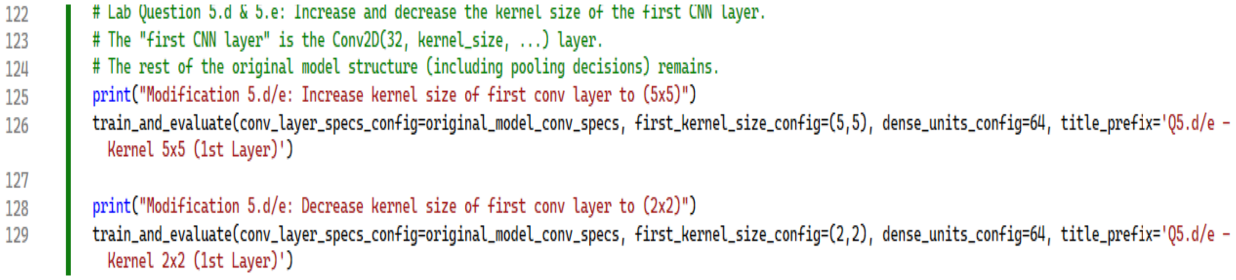
C.Output:



```
762/762 - 13s - 17ms/step - accuracy: 0.6675 - loss: 0.9635 - val_accuracy: 0.6400 - val_loss: 1.0755
Q5.c - Only 1st Conv Layer - Test accuracy: 0.6183
```

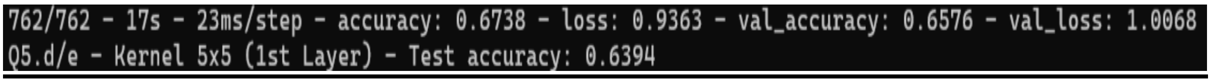


D. and E. Code:

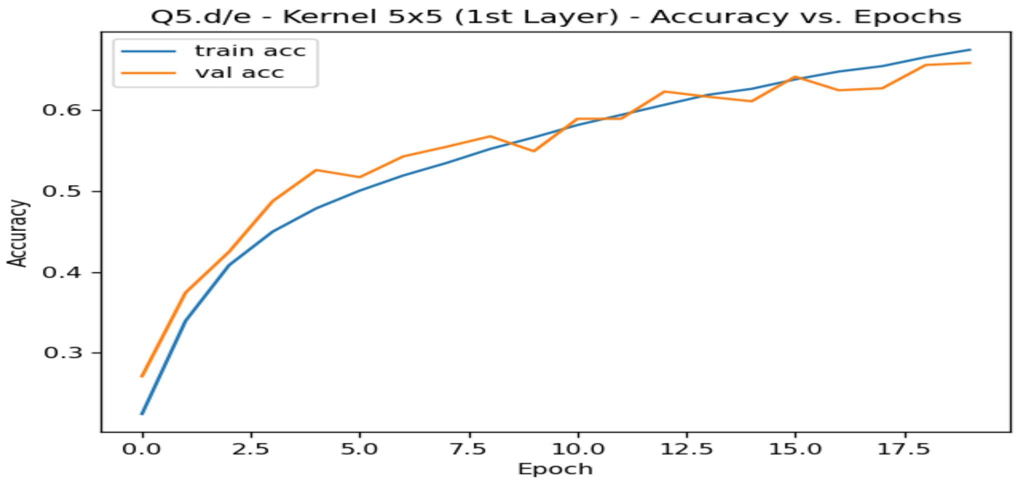


```
# Lab Question 5.d & 5.e: Increase and decrease the kernel size of the first CNN layer.
# The "first CNN layer" is the Conv2D(32, kernel_size, ...) layer.
# The rest of the original model structure (including pooling decisions) remains.
print("Modification 5.d/e: Increase kernel size of first conv layer to (5x5)")
train_and_evaluate(conv_layer_specs_config=original_model_conv_specs, first_kernel_size_config=(5,5), dense_units_config=64, title_prefix='Q5.d/e -
  Kernel 5x5 (1st Layer)')

print("Modification 5.d/e: Decrease kernel size of first conv layer to (2x2)")
train_and_evaluate(conv_layer_specs_config=original_model_conv_specs, first_kernel_size_config=(2,2), dense_units_config=64, title_prefix='Q5.d/e -
  Kernel 2x2 (1st Layer)')
```

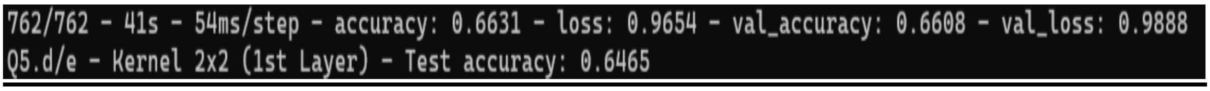
D.Output:



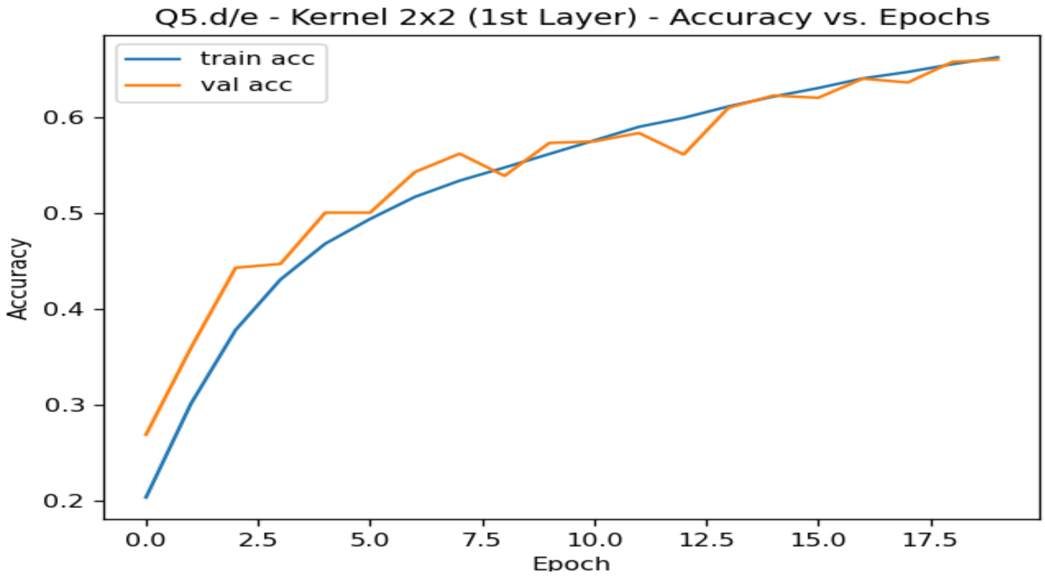
```
762/762 - 17s - 23ms/step - accuracy: 0.6738 - loss: 0.9363 - val_accuracy: 0.6576 - val_loss: 1.0068
Q5.d/e - Kernel 5x5 (1st Layer) - Test accuracy: 0.6394
```



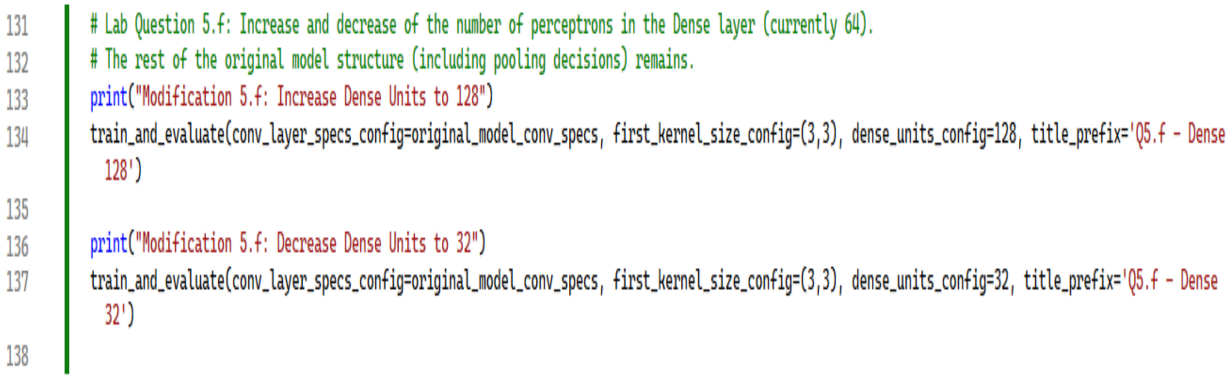
E.Output:



```
762/762 - 41s - 54ms/step - accuracy: 0.6631 - loss: 0.9654 - val_accuracy: 0.6608 - val_loss: 0.9888
Q5.d/e - Kernel 2x2 (1st Layer) - Test accuracy: 0.6465
```

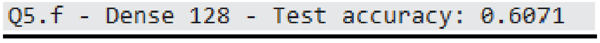


F.Code:

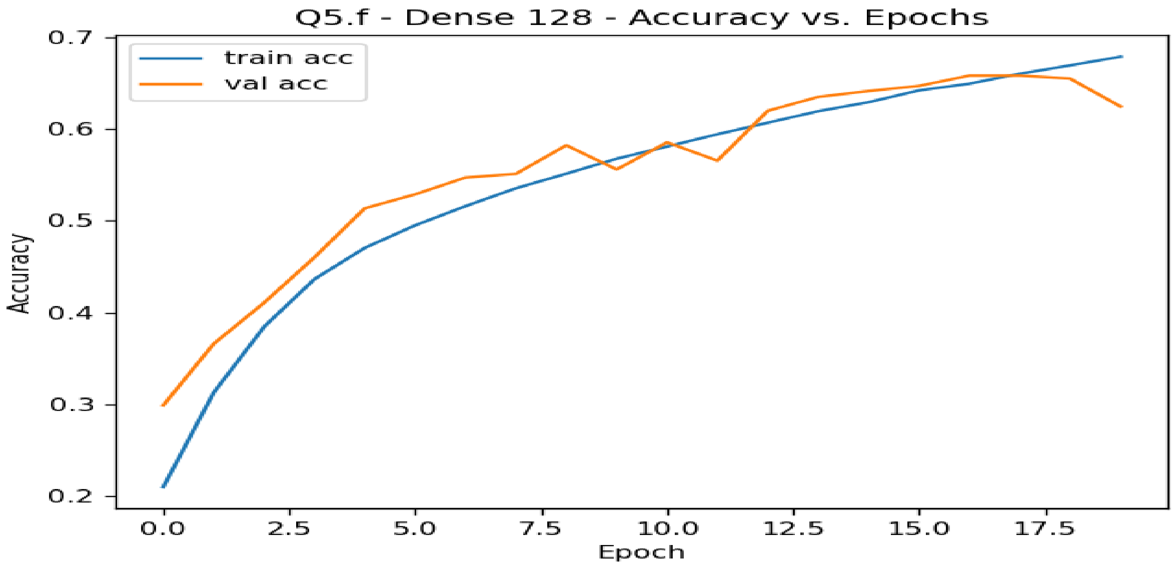


```
# Lab Question 5.f: Increase and decrease of the number of perceptrons in the Dense layer (currently 64).
# The rest of the original model structure (including pooling decisions) remains.
print("Modification 5.f: Increase Dense Units to 128")
train_and_evaluate(conv_layer_specs_config=original_model_conv_specs, first_kernel_size_config=(3,3), dense_units_config=128, title_prefix='Q5.f - Dense
  128')

print("Modification 5.f: Decrease Dense Units to 32")
train_and_evaluate(conv_layer_specs_config=original_model_conv_specs, first_kernel_size_config=(3,3), dense_units_config=32, title_prefix='Q5.f - Dense
  32')

```

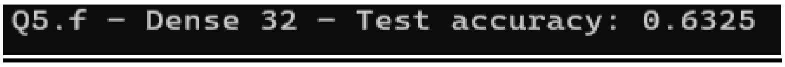
F.Output Increase:



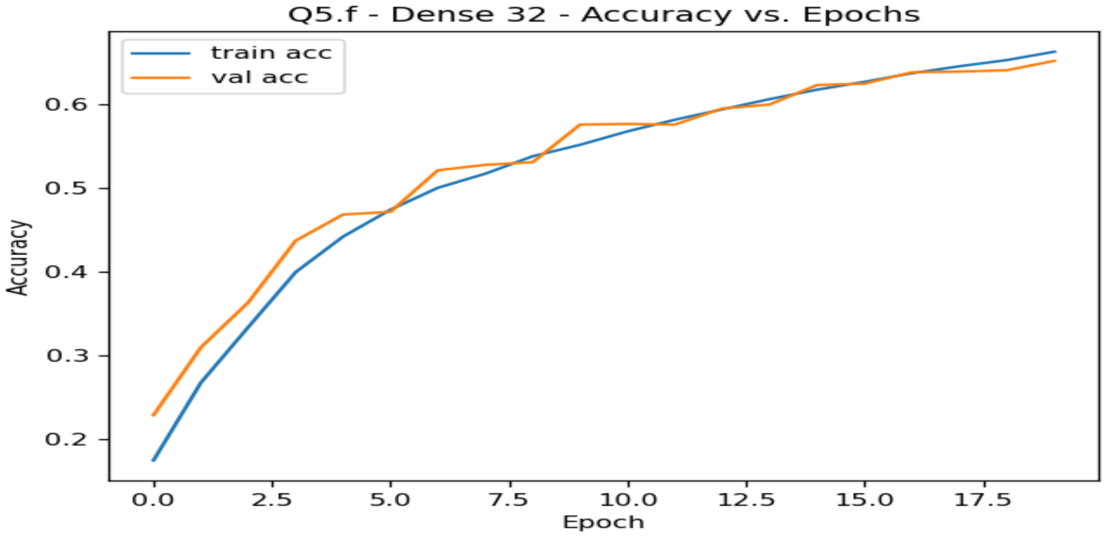
```
Q5.f - Dense 128 - Test accuracy: 0.6071
```



F.Output Decrease:



```
Q5.f – Dense 32 – Test accuracy: 0.6325
```



Q5.f - Dense 32 - Accuracy vs. Epochs
train acc
val acc
Accuracy
Epoch