Advanced Machine Learning (BA-64061)

ASSIGNMENT - 2

NEURAL NETWORKS

INTRODUCTION:

The objective of this assignment is to explore the impact of structural and hyper-parameter changes of a feed-forward neural network on performance on the IMDB movie-review sentiment dataset.
On top of the in-class baseline, several modifications to the model were implemented to examine network depth, hidden-unit size, loss function, activation function, and regularization.
The final objective is to identify a setting that maximizes the validation accuracy without compromising the generalization on unknown data.

METHODOLOGY:

DATASET
Source: Keras IMDB dataset containing 25 000 positive or negative film reviews.
Input representation: Integer-encoded words (vocabulary = top 10 000 most frequent tokens).
Sequence length: Reviews padded or truncated to 500 tokens.

Split: 80 % training, 20 % validation, along with an independent test set.

MODEL ARCHITECTURE

All models have the same general structure:

```
Embedding(10 000 → 64)
GlobalAveragePooling1D
Dense(hidden layers × units, activation)
Dense(1, activation='sigmoid')
```

TRAINING SETUP

- Optimizer: Adam (default learning rate)
- Batch size: 512 Epochs: 20
- Early Stopping on validation loss (patience = 3)
- Metrics: Accuracy and loss

EXPERIMENTS CONDUCTED

- Hidden layers: 1, 2 (baseline), 3
- Hidden units: 16, 32, 64
- Loss functions: `binary_crossentropy` vs `mse`
- Activations: `relu` vs `tanh`
- Regularization: Dropout (0.5), L2 (λ = 1e-4), Batch Normalization

  All of the runs have used early stopping and identical random seed for fairness and accuracy.

RESULTS:

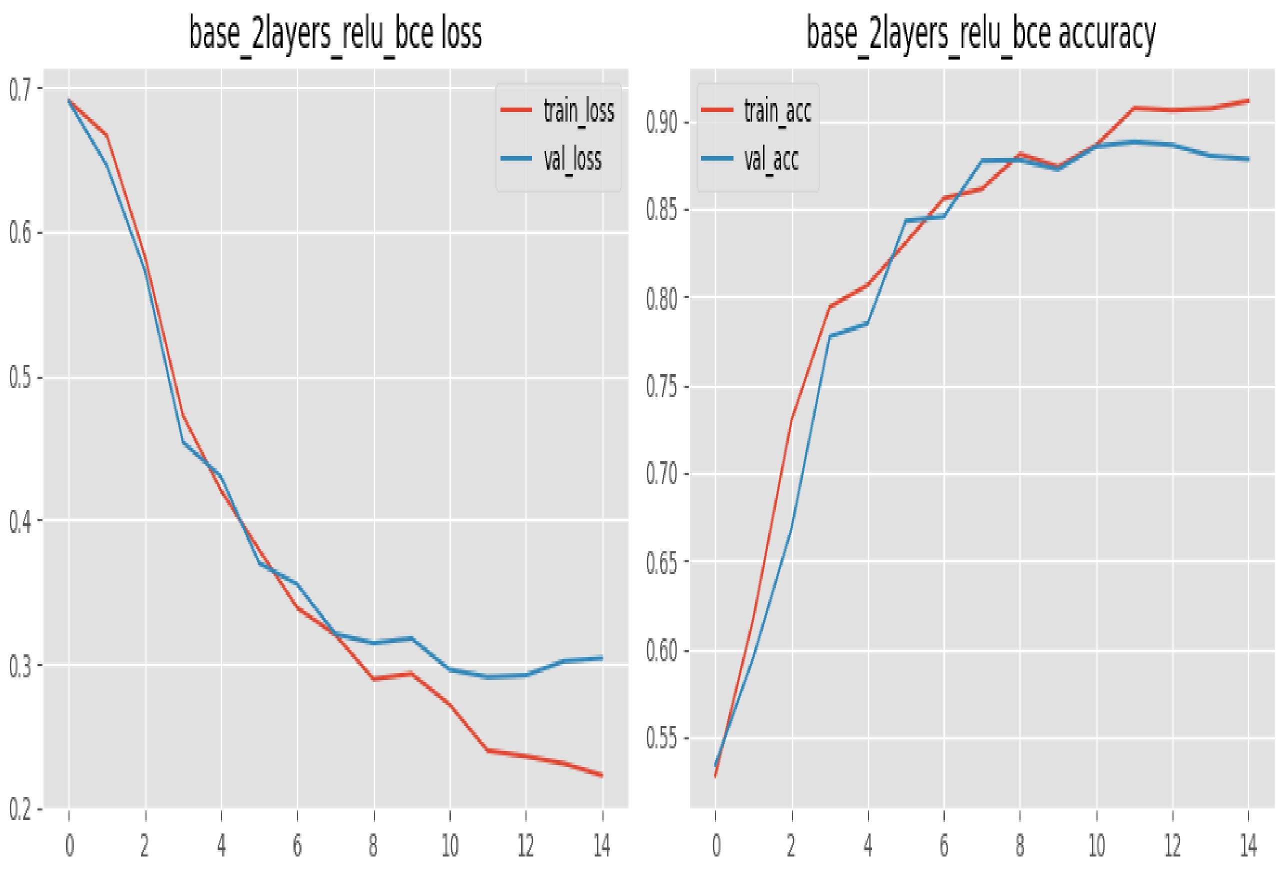
Quantitative Summary

Experiment	Val Acc	Test Acc	Val Loss	Test Loss
three_layer_64	**0.8928**	**0.8845**	0.2795	0.2879
units_64	0.8900	0.8828	0.2872	0.2944
units_32	0.8894	0.8824	0.2877	0.2952
tanh_activation	0.8884	0.8828	0.2925	0.2999
base_2layers_relu_bce	0.8878	0.8811	0.2909	0.2984
mse_loss	0.8834	0.8786	**0.0889**	**0.0916**
one_layer_16	0.8800	0.8750	0.2943	0.3013
dropout_0.5	0.8794	0.8716	0.3057	0.3148
l2_reg_1e-4	0.8620	0.8598	0.3427	0.3458

Visual Results

Baseline Model:

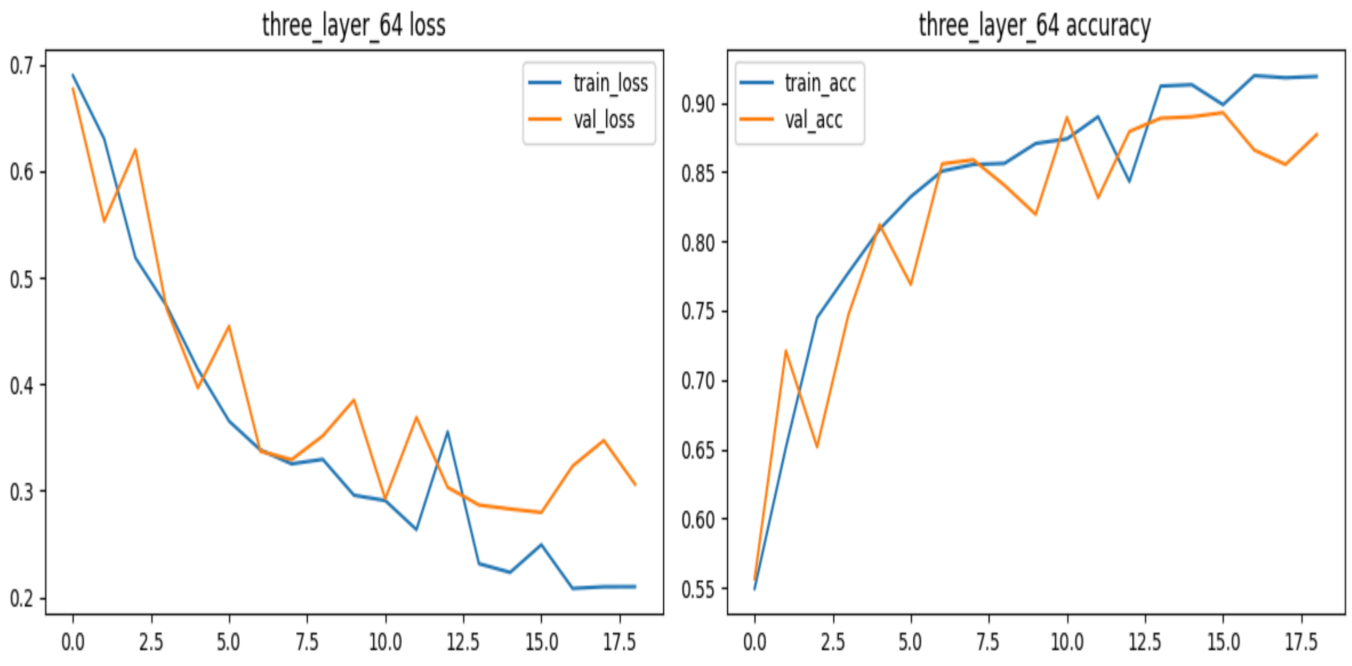
This plot demonstrates the training and validation loss and accuracy of the baseline neural network with two hidden layers of 32 ReLU units. The model converges well with little overfitting, with stable validation accuracy of around 0.88. Validation and training curves are close to each other, indicating good generalization.



Training and Validation Accuracy of the Baseline Model (Two Hidden Layers, ReLU, Binary Cross-Entropy)

Best Performing Model:

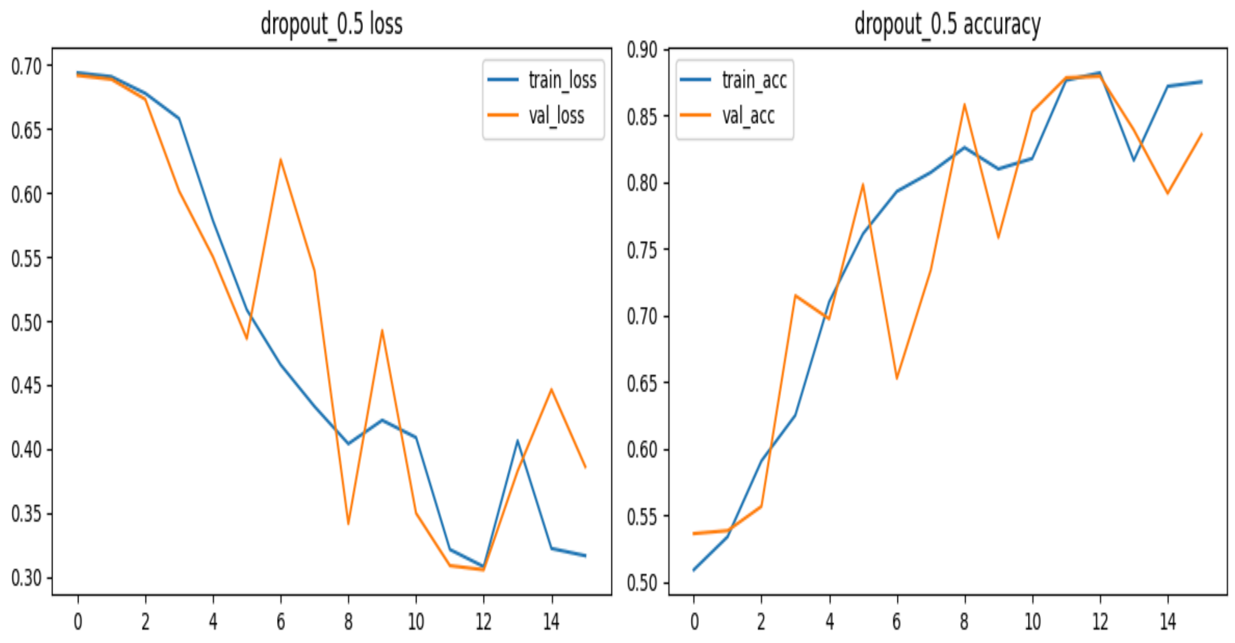
The highest validation accuracy (0.8928) and test accuracy (0.8845) were obtained by the three-layer setup of 64 ReLU units. The validation curve oscillates a bit more than the baseline, but that's just because the model is more complex. However, the model generalizes very well with no severe overfitting.



Three-Layer Model with 64 ReLU Units Performance

Regularized Model (Dropout 0.5):

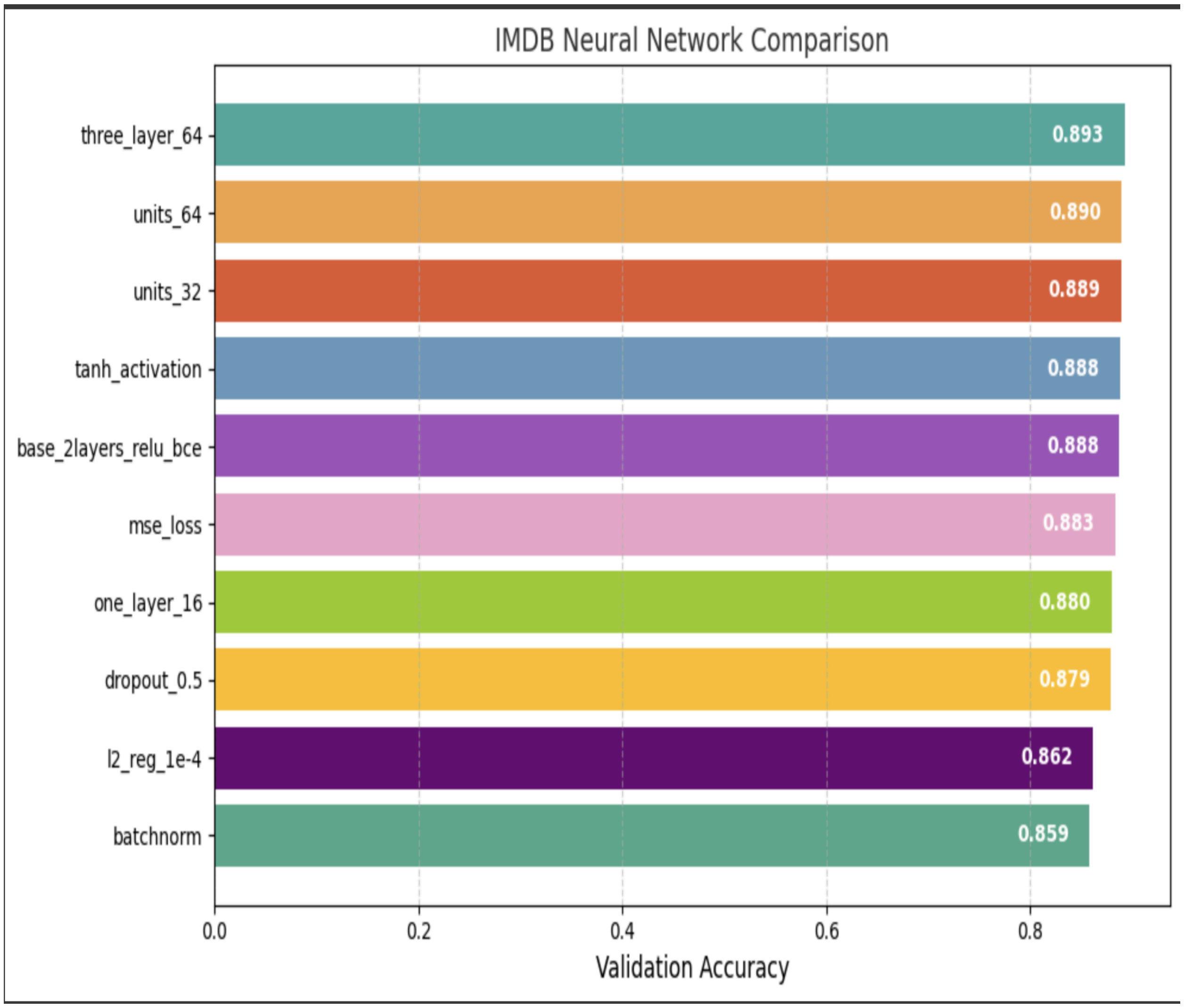
The dropout model (rate = 0.5) consists of smoother validation curves with less overfitting but slightly lower overall accuracy. Dropout randomly turns off neurons during training, preventing reliance on specific features and increasing robustness but at the cost of small loss of performance.



Effect of Dropout Regularization on Training Stability

Comparison Across All Experiments:

This bar chart recapitulates validation accuracy for all ten model configurations attempted. The three-layer 64-unit model achieved the best validation accuracy, very closely followed by the 64-unit and 32-unit two-layer models. The batch-normalized and regularized models were somewhat worse, revealing the trade-off between complexity and generalization.



Comparative Validation Accuracy of All Model Variants

DISCUSSION:

Network Depth
Increasing from one to three hidden layers improved validation accuracy from 0.88 to 0.893.
The three-layer network learned richer patterns with little overfitting but at the cost of longer training time.

Hidden Units
Increasing to 64 units resulted in a small boost in performance (≈ 0.002 accuracy improvement) at a small increase in loss, indicating higher variance.

Loss Function
Substituting binary_crossentropy with mse provided extremely small numeric loss values but a minor dip in accuracy, confirming that BCE is more suitable for binary classification.

Activation Function
relu performed better than tanh by about 0.5 % accuracy. Tanh showed slower convergence and slightly higher loss, as should be expected with ReLU's resistance to vanishing gradients.

Regularization
Dropout (0.5): Reduced overfitting but lowered maximum accuracy to 0.879.
L2 (1e-4): Provided negligible stability gain but no accuracy improvement.
Batch Normalization: Did not do well because the network was shallow to start with.

Overall, the three-layer ReLU model with 64 units and BCE loss did best in terms of accuracy vs. generalization.

CONCLUSION:

This exercise demonstrates the influence of minor architectural and hyper-parameter changes on neural-network performance.
The best configuration—three hidden layers with 64 ReLU units—had test accuracy 0.8845 and validation accuracy 0.8928.
While deeper networks yielded incremental gains, the basic two-layer network worked well and was effective.
Regularization aided stability but did not enhance accuracy.
ReLU activation and binary cross-entropy loss worked best for this task.

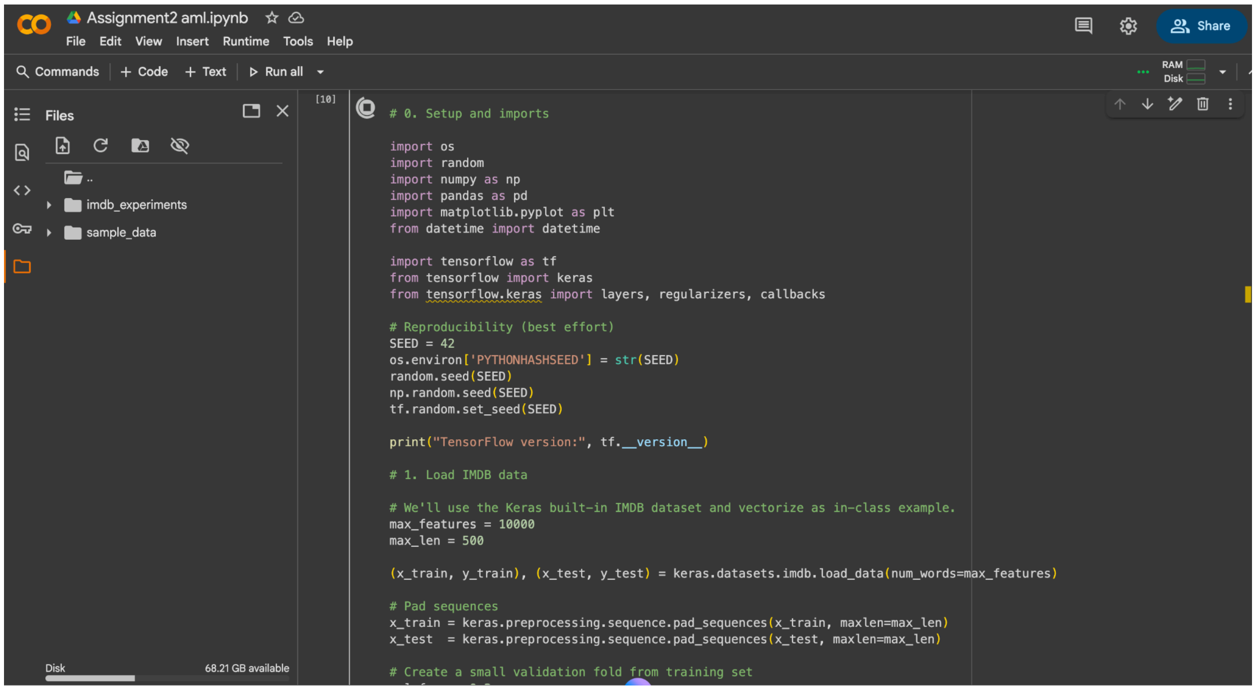
Further steps can involve sequence-aware models such as 1-D CNN or LSTM architecture for enhancement.

REFERENCES:

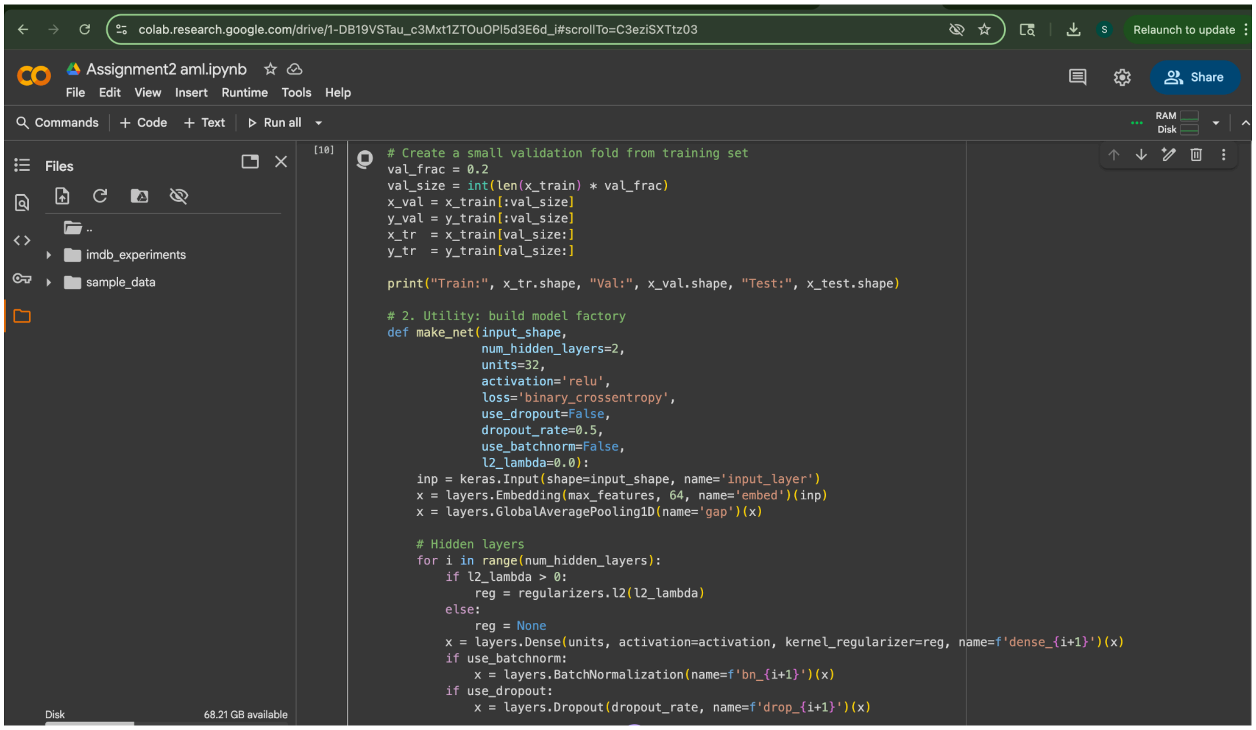
1. Keras Documentation – IMDB Dataset and Sequential Models
2. Chollet, F. *(2018). Deep Learning with Python.* Manning Publications.
3. Course Lecture Notes – Advanced Machine Learning (BA-64061)

Appendix:

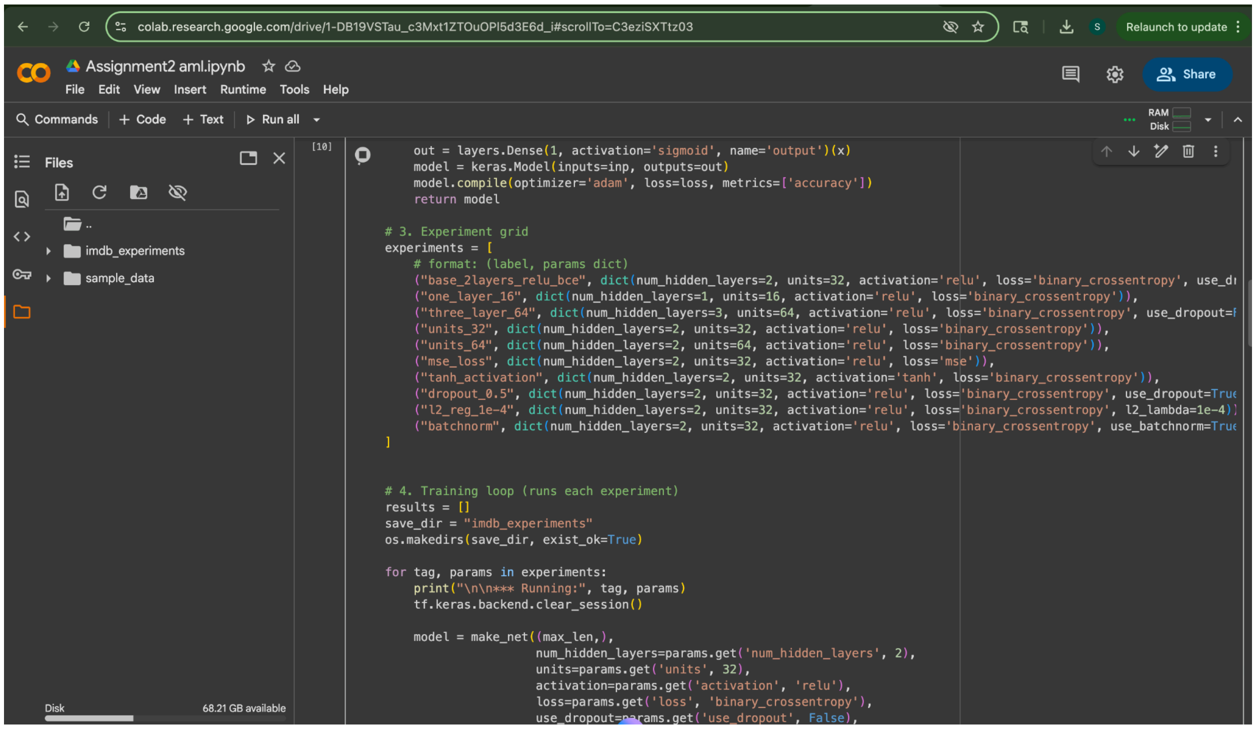
- Python version: 3.10   TensorFlow: 2.x
- Training executed in Google Colab GPU runtime.
- GitHub Repository link: *https://github.com/sathvika-vegiraju/svegira1_MachineLearning.git*



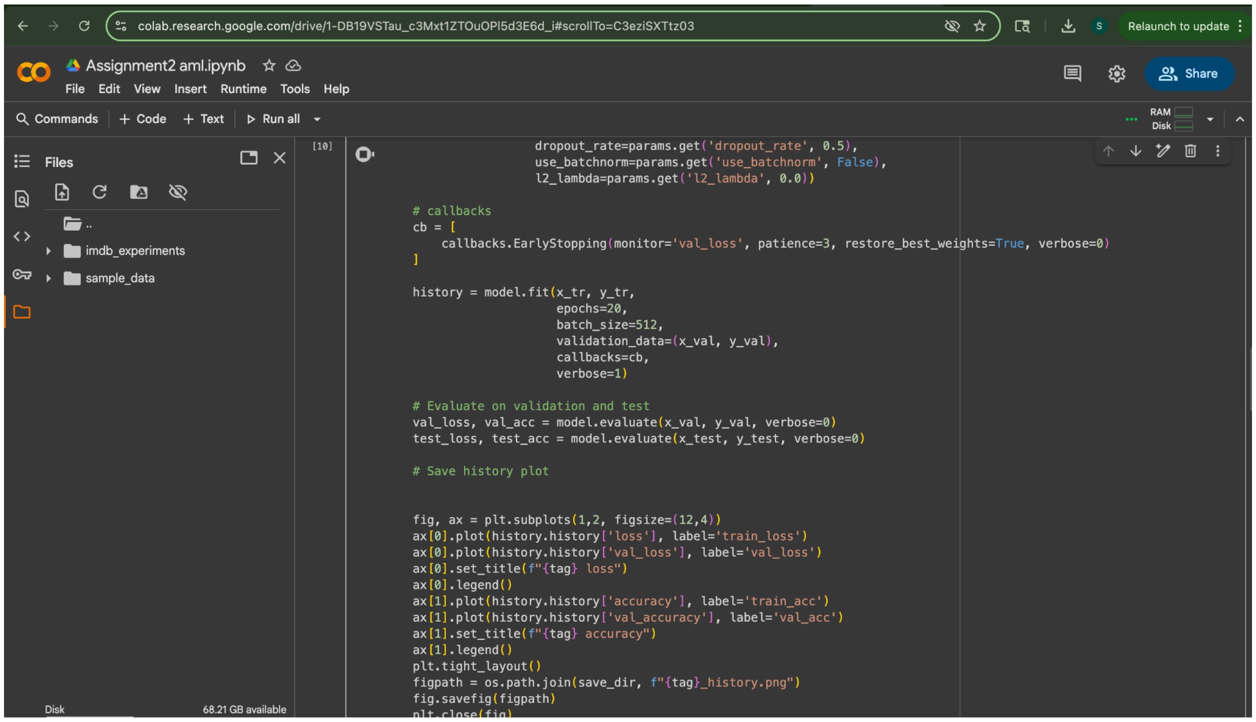
```python
# 0. Setup and imports

import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers, callbacks

# Reproducibility (best effort)
SEED = 42
os.environ['PYTHONHASHSEED'] = str(SEED)
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

print("TensorFlow version:", tf.__version__)

# 1. Load IMDB data

# We'll use the Keras built-in IMDB dataset and vectorize as in-class example.
max_features = 10000
max_len = 500

(x_train, y_train), (x_test, y_test) = keras.datasets.imdb.load_data(num_words=max_features)

# Pad sequences
x_train = keras.preprocessing.sequence.pad_sequences(x_train, maxlen=max_len)
x_test  = keras.preprocessing.sequence.pad_sequences(x_test, maxlen=max_len)

# Create a small validation fold from training set
```


colab.research.google.com/drive/1-DB19VSTau_c3Mxt1ZTOuOPl5d3E6d_i#scrollTo=C3eziSXTtz03
Relaunch to update
Assignment2 aml.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Files
imdb_experiments
sample_data
Disk 68.21 GB available
[10]

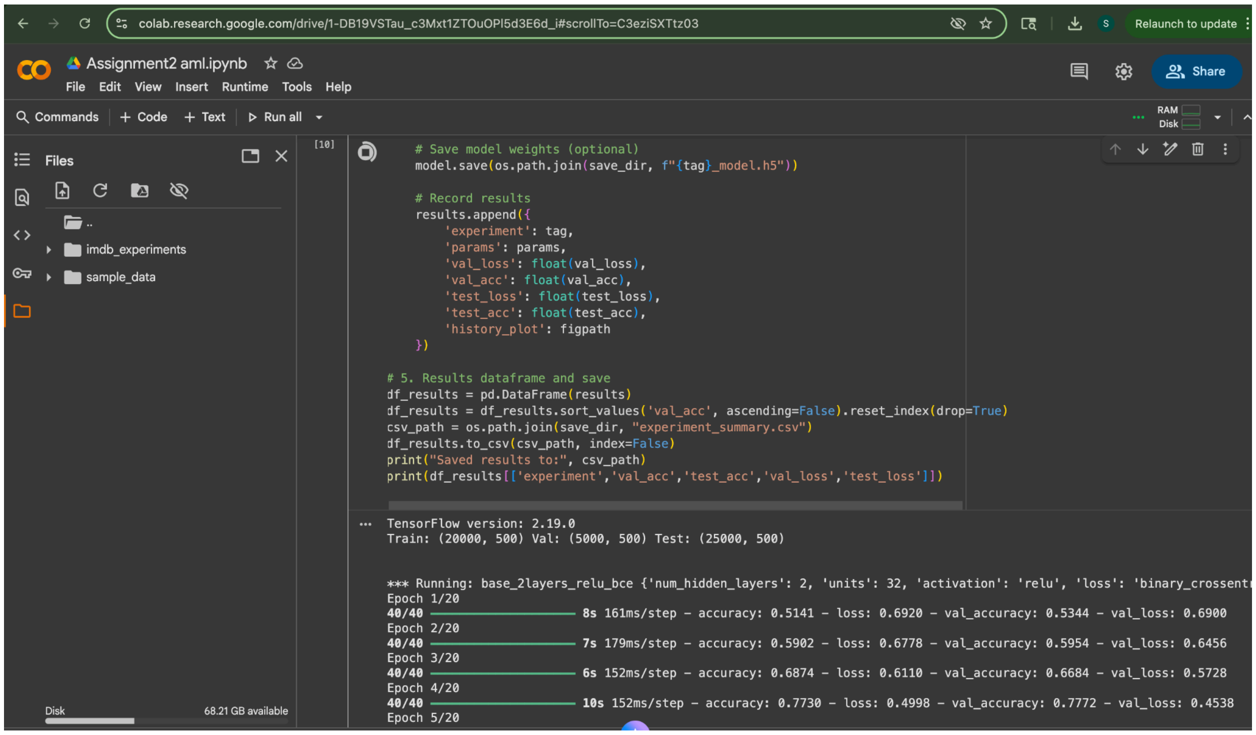
```python
# Create a small validation fold from training set
val_frac = 0.2
val_size = int(len(x_train) * val_frac)
x_val = x_train[:val_size]
y_val = y_train[:val_size]
x_tr  = x_train[val_size:]
y_tr  = y_train[val_size:]

print("Train:", x_tr.shape, "Val:", x_val.shape, "Test:", x_test.shape)

# 2. Utility: build model factory
def make_net(input_shape,
             num_hidden_layers=2,
             units=32,
             activation='relu',
             loss='binary_crossentropy',
             use_dropout=False,
             dropout_rate=0.5,
             use_batchnorm=False,
             l2_lambda=0.0):
    inp = keras.Input(shape=input_shape, name='input_layer')
    x = layers.Embedding(max_features, 64, name='embed')(inp)
    x = layers.GlobalAveragePooling1D(name='gap')(x)

    # Hidden layers
    for i in range(num_hidden_layers):
        if l2_lambda > 0:
            reg = regularizers.l2(l2_lambda)
        else:
            reg = None
        x = layers.Dense(units, activation=activation, kernel_regularizer=reg, name=f'dense_{i+1}')(x)
        if use_batchnorm:
            x = layers.BatchNormalization(name=f'bn_{i+1}')(x)
        if use_dropout:
            x = layers.Dropout(dropout_rate, name=f'drop_{i+1}')(x)
```



```python
    out = layers.Dense(1, activation='sigmoid', name='output')(x)
    model = keras.Model(inputs=inp, outputs=out)
    model.compile(optimizer='adam', loss=loss, metrics=['accuracy'])
    return model

# 3. Experiment grid
experiments = [
    # format: (label, params dict)
    ("base_2layers_relu_bce", dict(num_hidden_layers=2, units=32, activation='relu', loss='binary_crossentropy', use_d
    ("one_layer_16", dict(num_hidden_layers=1, units=16, activation='relu', loss='binary_crossentropy')),
    ("three_layer_64", dict(num_hidden_layers=3, units=64, activation='relu', loss='binary_crossentropy', use_dropout=
    ("units_32", dict(num_hidden_layers=2, units=32, activation='relu', loss='binary_crossentropy')),
    ("units_64", dict(num_hidden_layers=2, units=64, activation='relu', loss='binary_crossentropy')),
    ("mse_loss", dict(num_hidden_layers=2, units=32, activation='relu', loss='mse')),
    ("tanh_activation", dict(num_hidden_layers=2, units=32, activation='tanh', loss='binary_crossentropy')),
    ("dropout_0.5", dict(num_hidden_layers=2, units=32, activation='relu', loss='binary_crossentropy', use_dropout=True
    ("l2_reg_1e-4", dict(num_hidden_layers=2, units=32, activation='relu', loss='binary_crossentropy', l2_lambda=1e-4))
    ("batchnorm", dict(num_hidden_layers=2, units=32, activation='relu', loss='binary_crossentropy', use_batchnorm=True
]


# 4. Training loop (runs each experiment)
results = []
save_dir = "imdb_experiments"
os.makedirs(save_dir, exist_ok=True)

for tag, params in experiments:
    print("\n\n*** Running:", tag, params)
    tf.keras.backend.clear_session()

    model = make_net((max_len,),
                     num_hidden_layers=params.get('num_hidden_layers', 2),
                     units=params.get('units', 32),
                     activation=params.get('activation', 'relu'),
                     loss=params.get('loss', 'binary_crossentropy'),
                     use_dropout=params.get('use_dropout', False),
```



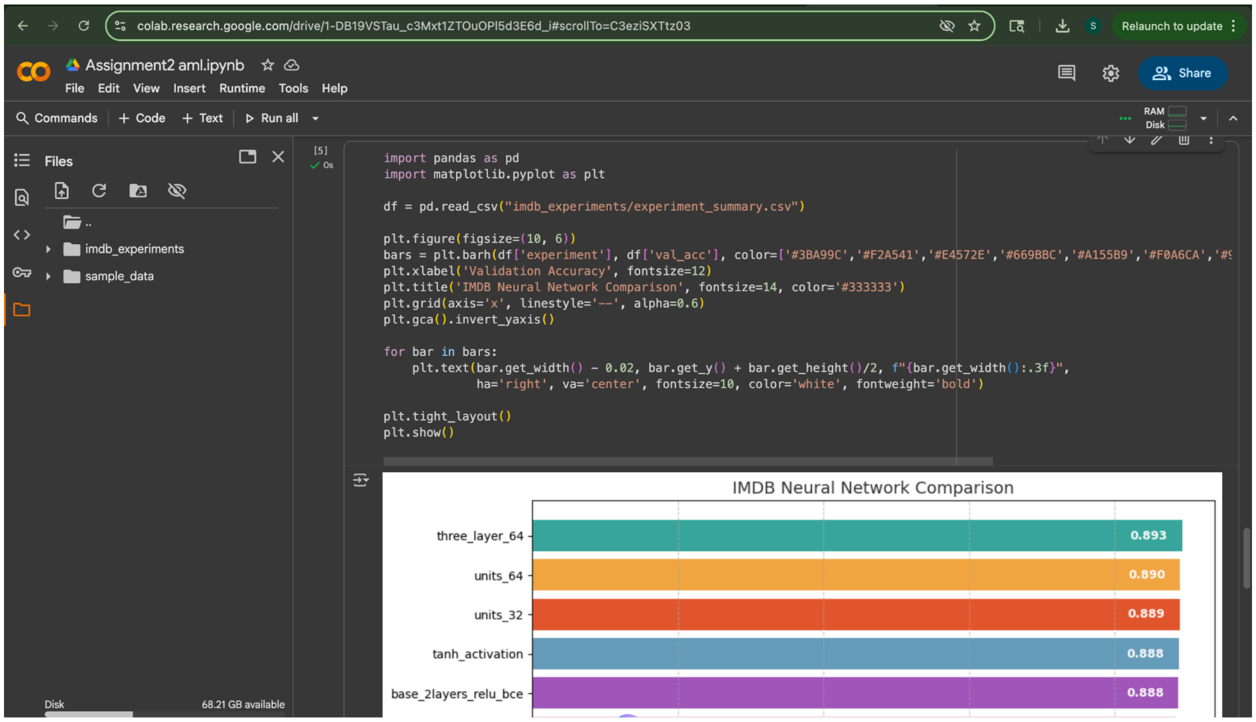
```python
                     dropout_rate=params.get('dropout_rate', 0.5),
                     use_batchnorm=params.get('use_batchnorm', False),
                     l2_lambda=params.get('l2_lambda', 0.0))

    # callbacks
    cb = [
        callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True, verbose=0)
    ]

    history = model.fit(x_tr, y_tr,
                        epochs=20,
                        batch_size=512,
                        validation_data=(x_val, y_val),
                        callbacks=cb,
                        verbose=1)

    # Evaluate on validation and test
    val_loss, val_acc = model.evaluate(x_val, y_val, verbose=0)
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)

    # Save history plot


    fig, ax = plt.subplots(1,2, figsize=(12,4))
    ax[0].plot(history.history['loss'], label='train_loss')
    ax[0].plot(history.history['val_loss'], label='val_loss')
    ax[0].set_title(f"{tag} loss")
    ax[0].legend()
    ax[1].plot(history.history['accuracy'], label='train_acc')
    ax[1].plot(history.history['val_accuracy'], label='val_acc')
    ax[1].set_title(f"{tag} accuracy")
    ax[1].legend()
    plt.tight_layout()
    figpath = os.path.join(save_dir, f"{tag}_history.png")
    fig.savefig(figpath)
    plt.close(fig)
```



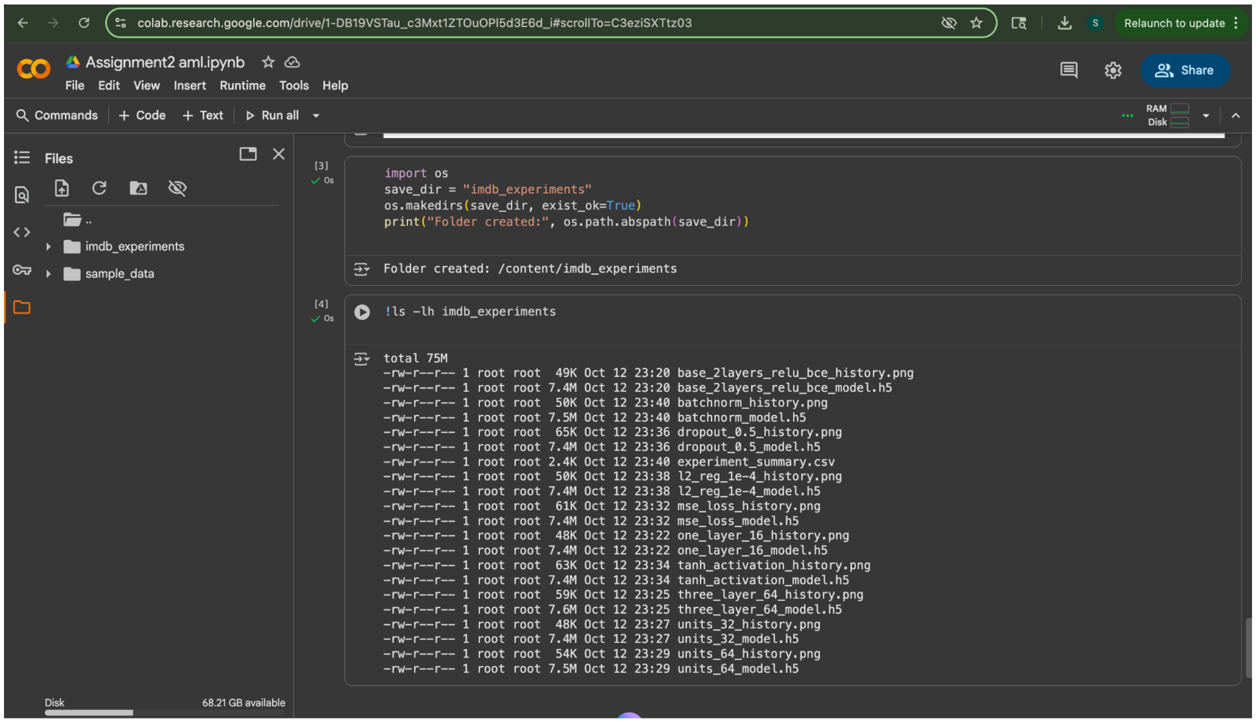
colab.research.google.com/drive/1-DB19VSTau_c3Mxt1ZTOuOPI5d3E6d_i#scrollTo=C3eziSXTtz03
Relaunch to update
Assignment2 aml.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text ▷ Run all
RAM
Disk
Files
..
imdb_experiments
sample_data
Disk 68.21 GB available
[10]

```
        # Save model weights (optional)
        model.save(os.path.join(save_dir, f"{tag}_model.h5"))

        # Record results
        results.append({
            'experiment': tag,
            'params': params,
            'val_loss': float(val_loss),
            'val_acc': float(val_acc),
            'test_loss': float(test_loss),
            'test_acc': float(test_acc),
            'history_plot': figpath
        })

    # 5. Results dataframe and save
    df_results = pd.DataFrame(results)
    df_results = df_results.sort_values('val_acc', ascending=False).reset_index(drop=True)
    csv_path = os.path.join(save_dir, "experiment_summary.csv")
    df_results.to_csv(csv_path, index=False)
    print("Saved results to:", csv_path)
    print(df_results[['experiment','val_acc','test_acc','val_loss','test_loss']])
```

```
TensorFlow version: 2.19.0
Train: (20000, 500) Val: (5000, 500) Test: (25000, 500)


*** Running: base_2layers_relu_bce {'num_hidden_layers': 2, 'units': 32, 'activation': 'relu', 'loss': 'binary_crossent
Epoch 1/20
40/40 ━━━━━━━━━━━━━━━━━━━━ 8s 161ms/step - accuracy: 0.5141 - loss: 0.6920 - val_accuracy: 0.5344 - val_loss: 0.6900
Epoch 2/20
40/40 ━━━━━━━━━━━━━━━━━━━━ 7s 179ms/step - accuracy: 0.5902 - loss: 0.6778 - val_accuracy: 0.5954 - val_loss: 0.6456
Epoch 3/20
40/40 ━━━━━━━━━━━━━━━━━━━━ 6s 152ms/step - accuracy: 0.6874 - loss: 0.6110 - val_accuracy: 0.6684 - val_loss: 0.5728
Epoch 4/20
40/40 ━━━━━━━━━━━━━━━━━━━━ 10s 152ms/step - accuracy: 0.7730 - loss: 0.4998 - val_accuracy: 0.7772 - val_loss: 0.4538
Epoch 5/20
```



```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("imdb_experiments/experiment_summary.csv")

plt.figure(figsize=(10, 6))
bars = plt.barh(df['experiment'], df['val_acc'], color=['#3BA99C','#F2A541','#E4572E','#669BBC','#A155B9','#F0A6CA','#9
plt.xlabel('Validation Accuracy', fontsize=12)
plt.title('IMDB Neural Network Comparison', fontsize=14, color='#333333')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.gca().invert_yaxis()

for bar in bars:
    plt.text(bar.get_width() - 0.02, bar.get_y() + bar.get_height()/2, f"{bar.get_width():.3f}",
             ha='right', va='center', fontsize=10, color='white', fontweight='bold')

plt.tight_layout()
plt.show()
```


colab.research.google.com/drive/1-DB19VSTau_c3Mxt1ZTOuOPl5d3E6d_i#scrollTo=C3eziSXTtz03
Relaunch to update
Assignment2 aml.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Files
..
imdb_experiments
sample_data
Disk 68.21 GB available

```
[3]
import os
save_dir = "imdb_experiments"
os.makedirs(save_dir, exist_ok=True)
print("Folder created:", os.path.abspath(save_dir))

Folder created: /content/imdb_experiments
```

```
[4]
!ls -lh imdb_experiments

total 75M
-rw-r--r-- 1 root root  49K Oct 12 23:20 base_2layers_relu_bce_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:20 base_2layers_relu_bce_model.h5
-rw-r--r-- 1 root root  50K Oct 12 23:40 batchnorm_history.png
-rw-r--r-- 1 root root 7.5M Oct 12 23:40 batchnorm_model.h5
-rw-r--r-- 1 root root  65K Oct 12 23:36 dropout_0.5_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:36 dropout_0.5_model.h5
-rw-r--r-- 1 root root 2.4K Oct 12 23:40 experiment_summary.csv
-rw-r--r-- 1 root root  50K Oct 12 23:38 l2_reg_1e-4_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:38 l2_reg_1e-4_model.h5
-rw-r--r-- 1 root root  61K Oct 12 23:32 mse_loss_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:32 mse_loss_model.h5
-rw-r--r-- 1 root root  48K Oct 12 23:22 one_layer_16_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:22 one_layer_16_model.h5
-rw-r--r-- 1 root root  63K Oct 12 23:34 tanh_activation_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:34 tanh_activation_model.h5
-rw-r--r-- 1 root root  59K Oct 12 23:25 three_layer_64_history.png
-rw-r--r-- 1 root root 7.6M Oct 12 23:25 three_layer_64_model.h5
-rw-r--r-- 1 root root  48K Oct 12 23:27 units_32_history.png
-rw-r--r-- 1 root root 7.4M Oct 12 23:27 units_32_model.h5
-rw-r--r-- 1 root root  54K Oct 12 23:29 units_64_history.png
-rw-r--r-- 1 root root 7.5M Oct 12 23:29 units_64_model.h5
```