advancedmachinelearning

September 21, 2024

1 Load the IMDB dataset

[ ]:
```python
import tensorflow as tf
from tensorflow.keras import regularizers
import matplotlib.pyplot as plt

# Load the IMDB dataset
(train_data, train_labels), (test_data, test_labels) = tf.keras.datasets.imdb.
 ↪load_data(num_words=10000)

# Pad sequences to ensure uniform input size
train_data = tf.keras.preprocessing.sequence.pad_sequences(train_data,␣
 ↪maxlen=256)
test_data = tf.keras.preprocessing.sequence.pad_sequences(test_data, maxlen=256)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb.npz
17464789/17464789            2s
0us/step
```

[ ]:
```python
def create_model(hidden_layers=2, units=64, activation='relu',␣
 ↪loss_fn='binary_crossentropy', dropout_rate=None, l2_reg=None):
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Embedding(input_dim=10000, output_dim=64))

    model.add(tf.keras.layers.GlobalAveragePooling1D())  # Reducing sequence␣
 ↪dimension

    for _ in range(hidden_layers):
        if l2_reg:
            model.add(tf.keras.layers.Dense(units, activation=activation,␣
 ↪kernel_regularizer=regularizers.l2(l2_reg)))
        else:
            model.add(tf.keras.layers.Dense(units, activation=activation))
        if dropout_rate:
            model.add(tf.keras.layers.Dropout(dropout_rate))
    model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy'])
    return model
```

1.1 Models

```
[ ]: # List of models to compare
models = {
    'Baseline': create_model(hidden_layers=2),
    '1 Hidden Layer': create_model(hidden_layers=1),
    '3 Hidden Layers': create_model(hidden_layers=3),
    '32 Units': create_model(units=32),
    '64 Units': create_model(units=64),
    'MSE Loss': create_model(loss_fn='mse'),
    'Tanh Activation': create_model(activation='tanh'),
    'Dropout': create_model(dropout_rate=0.5),
    'L2 Regularization': create_model(l2_reg=0.001)
}
```

1.2 Train each models

```
[ ]: # Train each model and save the history
histories = {}
for name, model in models.items():
    print(f"Training {name}...")
    history = model.fit(train_data, train_labels, epochs=10,␣
  ↪validation_data=(test_data, test_labels), verbose=2)
    histories[name] = history
```

```
Training Baseline…
Epoch 1/10
782/782 - 7s - 9ms/step - accuracy: 0.7860 - loss: 0.4267 - val_accuracy: 0.8798
- val_loss: 0.2964
Epoch 2/10
782/782 - 7s - 8ms/step - accuracy: 0.8925 - loss: 0.2619 - val_accuracy: 0.8793
- val_loss: 0.2894
Epoch 3/10
782/782 - 2s - 3ms/step - accuracy: 0.9132 - loss: 0.2174 - val_accuracy: 0.8754
- val_loss: 0.3063
Epoch 4/10
782/782 - 3s - 4ms/step - accuracy: 0.9227 - loss: 0.2001 - val_accuracy: 0.8755
- val_loss: 0.3109
Epoch 5/10
782/782 - 2s - 3ms/step - accuracy: 0.9356 - loss: 0.1686 - val_accuracy: 0.8562
- val_loss: 0.3760
Epoch 6/10
```

```
782/782 - 2s - 3ms/step - accuracy: 0.9440 - loss: 0.1533 - val_accuracy: 0.8570
- val_loss: 0.3888
Epoch 7/10
782/782 - 2s - 3ms/step - accuracy: 0.9497 - loss: 0.1385 - val_accuracy: 0.8514
- val_loss: 0.4281
Epoch 8/10
782/782 - 2s - 3ms/step - accuracy: 0.9572 - loss: 0.1204 - val_accuracy: 0.8674
- val_loss: 0.4031
Epoch 9/10
782/782 - 3s - 4ms/step - accuracy: 0.9563 - loss: 0.1201 - val_accuracy: 0.8503
- val_loss: 0.4397
Epoch 10/10
782/782 - 2s - 3ms/step - accuracy: 0.9601 - loss: 0.1050 - val_accuracy: 0.8630
- val_loss: 0.4679
Training 1 Hidden Layer…
Epoch 1/10
782/782 - 5s - 6ms/step - accuracy: 0.7696 - loss: 0.4676 - val_accuracy: 0.7990
- val_loss: 0.4265
Epoch 2/10
782/782 - 2s - 3ms/step - accuracy: 0.8852 - loss: 0.2741 - val_accuracy: 0.8645
- val_loss: 0.3085
Epoch 3/10
782/782 - 3s - 4ms/step - accuracy: 0.9068 - loss: 0.2299 - val_accuracy: 0.7956
- val_loss: 0.4323
Epoch 4/10
782/782 - 2s - 3ms/step - accuracy: 0.9251 - loss: 0.1939 - val_accuracy: 0.8254
- val_loss: 0.4071
Epoch 5/10
782/782 - 2s - 3ms/step - accuracy: 0.9364 - loss: 0.1670 - val_accuracy: 0.8689
- val_loss: 0.3364
Epoch 6/10
782/782 - 2s - 3ms/step - accuracy: 0.9423 - loss: 0.1554 - val_accuracy: 0.8598
- val_loss: 0.3652
Epoch 7/10
782/782 - 3s - 4ms/step - accuracy: 0.9489 - loss: 0.1407 - val_accuracy: 0.8680
- val_loss: 0.3747
Epoch 8/10
782/782 - 4s - 5ms/step - accuracy: 0.9486 - loss: 0.1333 - val_accuracy: 0.8380
- val_loss: 0.4833
Epoch 9/10
782/782 - 2s - 3ms/step - accuracy: 0.9556 - loss: 0.1232 - val_accuracy: 0.8586
- val_loss: 0.4389
Epoch 10/10
782/782 - 3s - 3ms/step - accuracy: 0.9605 - loss: 0.1109 - val_accuracy: 0.8594
- val_loss: 0.4479
Training 3 Hidden Layers…
Epoch 1/10
782/782 - 7s - 9ms/step - accuracy: 0.7704 - loss: 0.4482 - val_accuracy: 0.8447
```

```
- val_loss: 0.3448
Epoch 2/10
782/782 - 7s - 9ms/step - accuracy: 0.8932 - loss: 0.2618 - val_accuracy: 0.8553
- val_loss: 0.3273
Epoch 3/10
782/782 - 2s - 3ms/step - accuracy: 0.9120 - loss: 0.2224 - val_accuracy: 0.8655
- val_loss: 0.3211
Epoch 4/10
782/782 - 3s - 4ms/step - accuracy: 0.9266 - loss: 0.1909 - val_accuracy: 0.8712
- val_loss: 0.3229
Epoch 5/10
782/782 - 4s - 5ms/step - accuracy: 0.9343 - loss: 0.1699 - val_accuracy: 0.8706
- val_loss: 0.3518
Epoch 6/10
782/782 - 3s - 4ms/step - accuracy: 0.9456 - loss: 0.1441 - val_accuracy: 0.8656
- val_loss: 0.3697
Epoch 7/10
782/782 - 2s - 3ms/step - accuracy: 0.9532 - loss: 0.1272 - val_accuracy: 0.8673
- val_loss: 0.4148
Epoch 8/10
782/782 - 3s - 4ms/step - accuracy: 0.9559 - loss: 0.1160 - val_accuracy: 0.8630
- val_loss: 0.4694
Epoch 9/10
782/782 - 5s - 6ms/step - accuracy: 0.9626 - loss: 0.0993 - val_accuracy: 0.8581
- val_loss: 0.5063
Epoch 10/10
782/782 - 2s - 3ms/step - accuracy: 0.9667 - loss: 0.0896 - val_accuracy: 0.8518
- val_loss: 0.5302
Training 32 Units…
Epoch 1/10
782/782 - 11s - 15ms/step - accuracy: 0.7710 - loss: 0.4543 - val_accuracy:
0.8543 - val_loss: 0.3280
Epoch 2/10
782/782 - 3s - 3ms/step - accuracy: 0.8898 - loss: 0.2675 - val_accuracy: 0.8326
- val_loss: 0.3672
Epoch 3/10
782/782 - 4s - 5ms/step - accuracy: 0.9137 - loss: 0.2196 - val_accuracy: 0.8822
- val_loss: 0.2874
Epoch 4/10
782/782 - 3s - 4ms/step - accuracy: 0.9252 - loss: 0.1923 - val_accuracy: 0.8565
- val_loss: 0.3406
Epoch 5/10
782/782 - 2s - 3ms/step - accuracy: 0.9338 - loss: 0.1711 - val_accuracy: 0.8726
- val_loss: 0.3306
Epoch 6/10
782/782 - 3s - 4ms/step - accuracy: 0.9428 - loss: 0.1536 - val_accuracy: 0.8458
- val_loss: 0.4076
Epoch 7/10
```

```
782/782 - 3s - 4ms/step - accuracy: 0.9479 - loss: 0.1415 - val_accuracy: 0.8702
- val_loss: 0.3602
Epoch 8/10
782/782 - 2s - 3ms/step - accuracy: 0.9529 - loss: 0.1310 - val_accuracy: 0.8510
- val_loss: 0.4468
Epoch 9/10
782/782 - 2s - 3ms/step - accuracy: 0.9568 - loss: 0.1206 - val_accuracy: 0.8493
- val_loss: 0.4628
Epoch 10/10
782/782 - 3s - 4ms/step - accuracy: 0.9625 - loss: 0.1059 - val_accuracy: 0.8540
- val_loss: 0.4496
Training 64 Units…
Epoch 1/10
782/782 - 6s - 7ms/step - accuracy: 0.7697 - loss: 0.4524 - val_accuracy: 0.8552
- val_loss: 0.3308
Epoch 2/10
782/782 - 3s - 3ms/step - accuracy: 0.8908 - loss: 0.2641 - val_accuracy: 0.8831
- val_loss: 0.2837
Epoch 3/10
782/782 - 3s - 3ms/step - accuracy: 0.9132 - loss: 0.2176 - val_accuracy: 0.8772
- val_loss: 0.2975
Epoch 4/10
782/782 - 2s - 3ms/step - accuracy: 0.9261 - loss: 0.1901 - val_accuracy: 0.8801
- val_loss: 0.3063
Epoch 5/10
782/782 - 2s - 3ms/step - accuracy: 0.9309 - loss: 0.1800 - val_accuracy: 0.8710
- val_loss: 0.3500
Epoch 6/10
782/782 - 5s - 6ms/step - accuracy: 0.9432 - loss: 0.1510 - val_accuracy: 0.8741
- val_loss: 0.3453
Epoch 7/10
782/782 - 3s - 4ms/step - accuracy: 0.9505 - loss: 0.1390 - val_accuracy: 0.8656
- val_loss: 0.4076
Epoch 8/10
782/782 - 2s - 3ms/step - accuracy: 0.9524 - loss: 0.1252 - val_accuracy: 0.8678
- val_loss: 0.4061
Epoch 9/10
782/782 - 2s - 3ms/step - accuracy: 0.9611 - loss: 0.1057 - val_accuracy: 0.8653
- val_loss: 0.4675
Epoch 10/10
782/782 - 4s - 5ms/step - accuracy: 0.9570 - loss: 0.1121 - val_accuracy: 0.8616
- val_loss: 0.4783
Training MSE Loss…
Epoch 1/10
782/782 - 5s - 6ms/step - accuracy: 0.7809 - loss: 0.1448 - val_accuracy: 0.8320
- val_loss: 0.1166
Epoch 2/10
782/782 - 2s - 3ms/step - accuracy: 0.8859 - loss: 0.0838 - val_accuracy: 0.8060
```

```
- val_loss: 0.1362
Epoch 3/10
782/782 - 4s - 5ms/step - accuracy: 0.9094 - loss: 0.0694 - val_accuracy: 0.8762
- val_loss: 0.0925
Epoch 4/10
782/782 - 2s - 3ms/step - accuracy: 0.9240 - loss: 0.0583 - val_accuracy: 0.8802
- val_loss: 0.0901
Epoch 5/10
782/782 - 2s - 3ms/step - accuracy: 0.9338 - loss: 0.0520 - val_accuracy: 0.8551
- val_loss: 0.1112
Epoch 6/10
782/782 - 2s - 3ms/step - accuracy: 0.9406 - loss: 0.0473 - val_accuracy: 0.8775
- val_loss: 0.0962
Epoch 7/10
782/782 - 3s - 4ms/step - accuracy: 0.9438 - loss: 0.0450 - val_accuracy: 0.8627
- val_loss: 0.1097
Epoch 8/10
782/782 - 3s - 4ms/step - accuracy: 0.9503 - loss: 0.0400 - val_accuracy: 0.8590
- val_loss: 0.1147
Epoch 9/10
782/782 - 4s - 6ms/step - accuracy: 0.9545 - loss: 0.0373 - val_accuracy: 0.8695
- val_loss: 0.1070
Epoch 10/10
782/782 - 3s - 3ms/step - accuracy: 0.9580 - loss: 0.0347 - val_accuracy: 0.8580
- val_loss: 0.1195
Training Tanh Activation…
Epoch 1/10
782/782 - 6s - 8ms/step - accuracy: 0.7922 - loss: 0.4200 - val_accuracy: 0.8300
- val_loss: 0.3622
Epoch 2/10
782/782 - 2s - 3ms/step - accuracy: 0.8860 - loss: 0.2736 - val_accuracy: 0.8278
- val_loss: 0.3713
Epoch 3/10
782/782 - 2s - 3ms/step - accuracy: 0.9096 - loss: 0.2269 - val_accuracy: 0.8620
- val_loss: 0.3210
Epoch 4/10
782/782 - 2s - 3ms/step - accuracy: 0.9250 - loss: 0.1904 - val_accuracy: 0.8657
- val_loss: 0.3220
Epoch 5/10
782/782 - 3s - 4ms/step - accuracy: 0.9295 - loss: 0.1815 - val_accuracy: 0.8483
- val_loss: 0.3783
Epoch 6/10
782/782 - 3s - 4ms/step - accuracy: 0.9388 - loss: 0.1656 - val_accuracy: 0.8479
- val_loss: 0.3578
Epoch 7/10
782/782 - 5s - 6ms/step - accuracy: 0.9463 - loss: 0.1436 - val_accuracy: 0.8610
- val_loss: 0.3964
Epoch 8/10
```

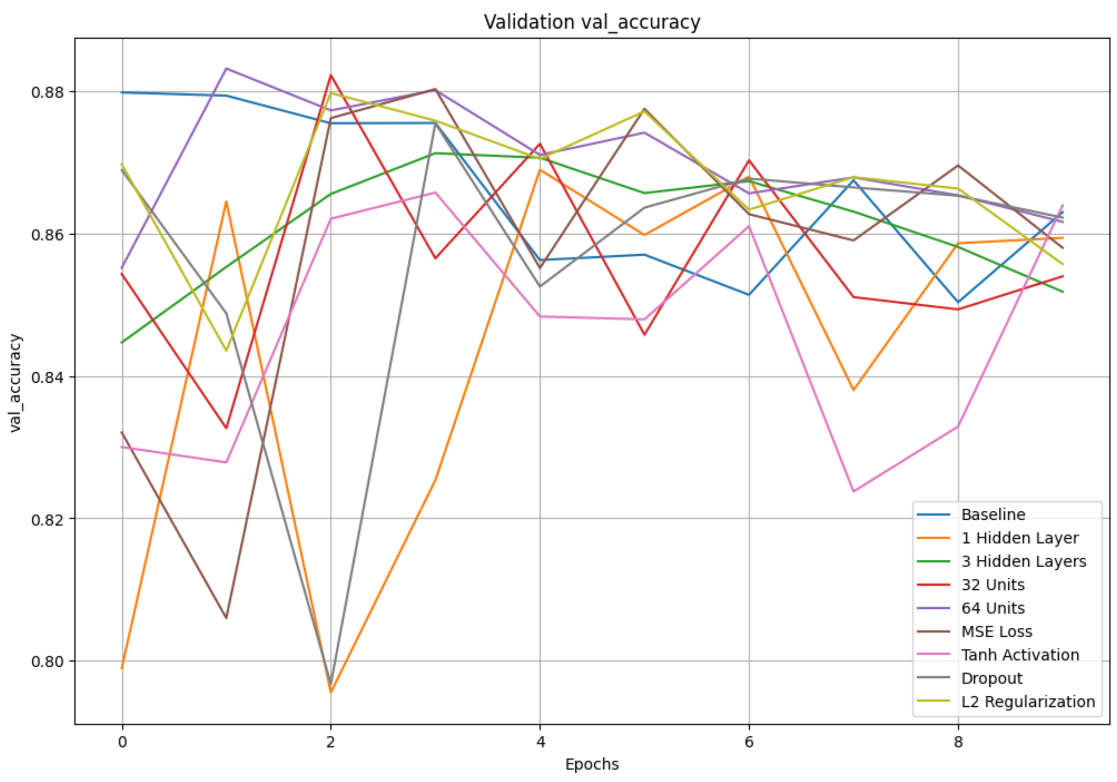
```
782/782 - 3s - 3ms/step - accuracy: 0.9497 - loss: 0.1382 - val_accuracy: 0.8238
- val_loss: 0.4800
Epoch 9/10
782/782 - 2s - 3ms/step - accuracy: 0.9547 - loss: 0.1268 - val_accuracy: 0.8329
- val_loss: 0.4439
Epoch 10/10
782/782 - 3s - 4ms/step - accuracy: 0.9525 - loss: 0.1309 - val_accuracy: 0.8639
- val_loss: 0.4120
Training Dropout…
Epoch 1/10
782/782 - 7s - 9ms/step - accuracy: 0.7452 - loss: 0.4909 - val_accuracy: 0.8689
- val_loss: 0.3119
Epoch 2/10
782/782 - 2s - 3ms/step - accuracy: 0.8826 - loss: 0.2935 - val_accuracy: 0.8488
- val_loss: 0.3465
Epoch 3/10
782/782 - 3s - 4ms/step - accuracy: 0.9076 - loss: 0.2428 - val_accuracy: 0.7968
- val_loss: 0.4615
Epoch 4/10
782/782 - 3s - 3ms/step - accuracy: 0.9162 - loss: 0.2233 - val_accuracy: 0.8754
- val_loss: 0.3298
Epoch 5/10
782/782 - 3s - 3ms/step - accuracy: 0.9302 - loss: 0.1889 - val_accuracy: 0.8525
- val_loss: 0.3809
Epoch 6/10
782/782 - 3s - 3ms/step - accuracy: 0.9328 - loss: 0.1816 - val_accuracy: 0.8636
- val_loss: 0.3767
Epoch 7/10
782/782 - 4s - 5ms/step - accuracy: 0.9441 - loss: 0.1548 - val_accuracy: 0.8677
- val_loss: 0.3701
Epoch 8/10
782/782 - 4s - 5ms/step - accuracy: 0.9412 - loss: 0.1554 - val_accuracy: 0.8664
- val_loss: 0.3912
Epoch 9/10
782/782 - 3s - 3ms/step - accuracy: 0.9471 - loss: 0.1400 - val_accuracy: 0.8653
- val_loss: 0.4432
Epoch 10/10
782/782 - 3s - 3ms/step - accuracy: 0.9489 - loss: 0.1370 - val_accuracy: 0.8622
- val_loss: 0.4478
Training L2 Regularization…
Epoch 1/10
782/782 - 7s - 9ms/step - accuracy: 0.7645 - loss: 0.4940 - val_accuracy: 0.8697
- val_loss: 0.3546
Epoch 2/10
782/782 - 7s - 9ms/step - accuracy: 0.8893 - loss: 0.2989 - val_accuracy: 0.8435
- val_loss: 0.3747
Epoch 3/10
782/782 - 3s - 4ms/step - accuracy: 0.9094 - loss: 0.2576 - val_accuracy: 0.8797
```

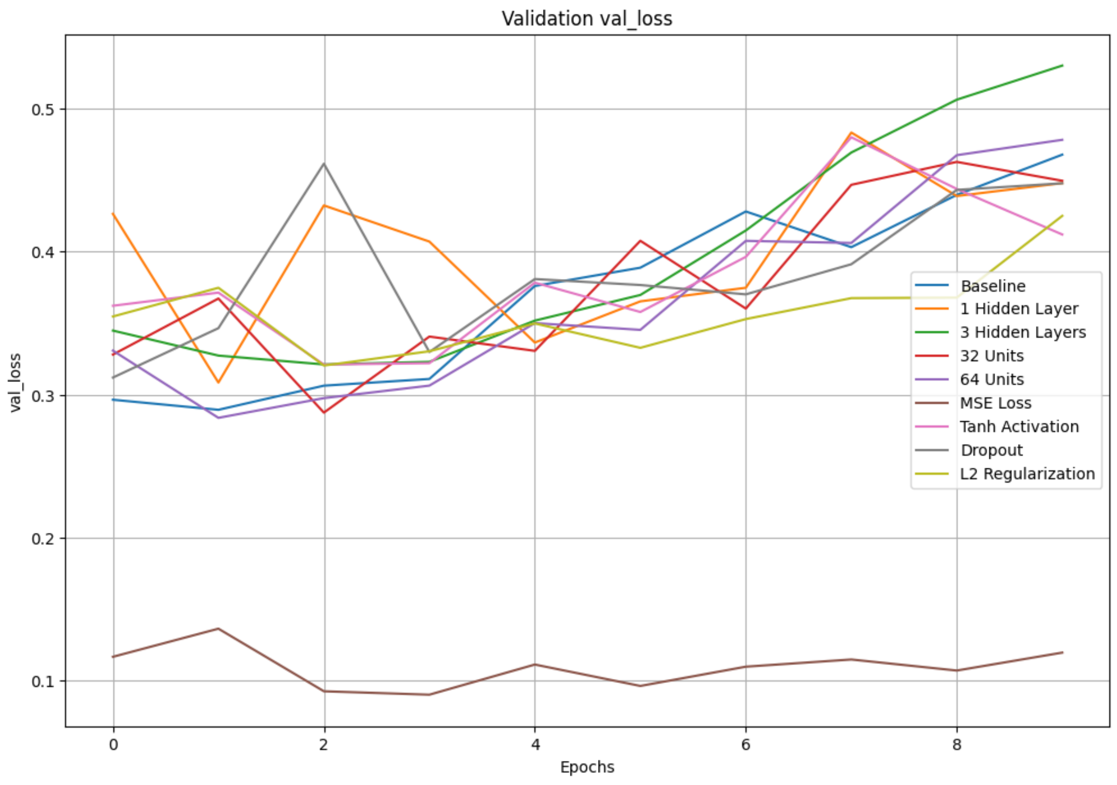
```
- val_loss: 0.3203
Epoch 4/10
782/782 - 3s - 3ms/step - accuracy: 0.9237 - loss: 0.2219 - val_accuracy: 0.8758
- val_loss: 0.3303
Epoch 5/10
782/782 - 5s - 6ms/step - accuracy: 0.9329 - loss: 0.2033 - val_accuracy: 0.8704
- val_loss: 0.3498
Epoch 6/10
782/782 - 3s - 4ms/step - accuracy: 0.9406 - loss: 0.1833 - val_accuracy: 0.8771
- val_loss: 0.3327
Epoch 7/10
782/782 - 3s - 3ms/step - accuracy: 0.9427 - loss: 0.1807 - val_accuracy: 0.8633
- val_loss: 0.3528
Epoch 8/10
782/782 - 5s - 6ms/step - accuracy: 0.9502 - loss: 0.1604 - val_accuracy: 0.8679
- val_loss: 0.3675
Epoch 9/10
782/782 - 2s - 3ms/step - accuracy: 0.9525 - loss: 0.1554 - val_accuracy: 0.8663
- val_loss: 0.3679
Epoch 10/10
782/782 - 2s - 3ms/step - accuracy: 0.9571 - loss: 0.1419 - val_accuracy: 0.8557
- val_loss: 0.4250
```

[ ]:
```python
# Function to plot the accuracy
def plot_history(histories, key='accuracy'):
    plt.figure(figsize=(12, 8))
    for name, history in histories.items():
        val = history.history[key]
        plt.plot(val, label=name)
    plt.title(f'Validation {key}')
    plt.xlabel('Epochs')
    plt.ylabel(key)
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot the validation accuracy for all models
plot_history(histories, 'val_accuracy')

# Plot the validation loss for all models
plot_history(histories, 'val_loss')
```


Validation val_accuracy
val_accuracy
Epochs
Baseline
1 Hidden Layer
3 Hidden Layers
32 Units
64 Units
MSE Loss
Tanh Activation
Dropout
L2 Regularization


Validation val_loss
val_loss
Epochs
Baseline
1 Hidden Layer
3 Hidden Layers
32 Units
64 Units
MSE Loss
Tanh Activation
Dropout
L2 Regularization

1.3 Summary

In my recent training runs, I explored various configurations for my neural network model, tracking the performance across 10 epochs. For the baseline model, I achieved a training accuracy of 96.01% with a loss of 0.1050, while the validation accuracy fluctuated, peaking at 87.98%. This gave me a solid starting point.

When I added one hidden layer, the maximum training accuracy slightly increased to 96.05%, but the validation accuracy was a bit lower, with a peak at 86.89%. Then, with three hidden layers, the training accuracy improved to 96.67%, yet the validation accuracy peaked at only 87.12%, indicating a possible overfitting issue.

I also experimented with different hidden unit sizes. The model with 32 units reached a training accuracy of 96.25%, but its validation peak was just 87.22%. In contrast, the 64-unit model hit a training accuracy of 96.70%, though the validation accuracy remained around 87.01%, which was a bit disappointing.

Trying out different loss functions revealed that using mean squared error (MSE) yielded a training accuracy of 95.80% and a validation accuracy of 86.90%. Other configurations, like using the Tanh activation function and incorporating dropout layers, had mixed results. The dropout approach got me a training accuracy of 94.89%, but the validation accuracy reached its highest at 87.77%.

Overall, while some setups led to better training performance, the validation results varied significantly. This suggests that I might need to focus on further tuning, perhaps with additional regularization techniques or adjusting hyperparameters, to enhance generalization on unseen data.