# Hyperparameter Tuning Results

## Experimental Setup

This experiment evaluates different methods for weighting ELMo embeddings in a text classification task. We compare three approaches:

1. **Trainable Lambdas**: The lambda values for combining ELMo layers are learned during training.
2. **Frozen Lambdas**: Predefined lambda values remain fixed throughout training.
3. **Learnable Function**: A neural network-based function learns to combine ELMo layers.

Each method was trained for three epochs with the same downstream classifier architecture. The final test accuracies and lambda values for the ELMo layers are reported.

## Results

### Trainable Lambdas

- **Final Test Accuracy**: **0.8832**
- **Best Test Accuracy**: **0.8832**
- **Final Lambda Values**: `[0.636, 0.149, 0.215]`
- **Epoch-wise Performance**:
  - Epoch 1: Train Loss = 0.6060, Train Acc = 0.8742, Test Acc = 0.8549
  - Epoch 2: Train Loss = 0.3518, Train Acc = 0.9087, Test Acc = 0.8779
  - Epoch 3: Train Loss = 0.2559, Train Acc = 0.9377, Test Acc = 0.8832
- **Confusion Matrix**: `confusion_matrix_trainable_lambda.png`

### Frozen Lambdas

- **Final Test Accuracy**: **0.8675**
- **Best Test Accuracy**: **0.8675**
- **Final Lambda Values**: `[0.079, 0.014, 0.907]`
- **Epoch-wise Performance**:
  - Epoch 1: Train Loss = 0.6099, Train Acc = 0.8461, Test Acc = 0.8315
  - Epoch 2: Train Loss = 0.4017, Train Acc = 0.8828, Test Acc = 0.8590
  - Epoch 3: Train Loss = 0.3447, Train Acc = 0.8909, Test Acc = 0.8675
- **Confusion Matrix**: `confusion_matrix_frozen_lambda.png`

PROF

### Learnable Function

- **Final Test Accuracy**: **0.8788**
- **Best Test Accuracy**: **0.8788**
- **Epoch-wise Performance**:
  - Epoch 1: Train Loss = 0.5849, Train Acc = 0.8654, Test Acc = 0.8515
  - Epoch 2: Train Loss = 0.3741, Train Acc = 0.8849, Test Acc = 0.8643
  - Epoch 3: Train Loss = 0.3342, Train Acc = 0.9014, Test Acc = 0.8788

- **Confusion Matrix**: `confusion_matrix_learnable_function.png`

### Comparison and Analysis

- **Best Method**: **Trainable Lambdas** achieved the highest test accuracy (**0.8832**), outperforming **Learnable Function (0.8788)** and **Frozen Lambdas (0.8675)**.
- **Interpretation of Lambda Values**:
  - Trainable lambda values suggest that the first layer contributes the most (**0.636**), while the second and third layers have smaller but significant contributions.
  - The frozen lambda method heavily relied on the third layer (**0.907**), which may have resulted in slightly lower performance.
- **Effect of Training**:
  - Trainable lambdas improved over epochs, whereas frozen lambda values limited the model's ability to adapt.
  - The learnable function also improved performance but was slightly less effective than direct trainable weights.

## Comparison with Other Embedding Techniques

To further evaluate embedding effectiveness, we compare ELMo with traditional embedding methods: **SVD, Skip-gram, and CBOW**. Each method is used in the same classification model, and performance is assessed using accuracy, precision, recall, F1-score, and confusion matrices.

| Embedding Method | Test Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| **ELMo (Trainable Lambdas)** | **0.8832** | **0.89** | **0.88** | **0.885** |
| **ELMo (Learnable Function)** | 0.8788 | 0.88 | 0.88 | 0.879 |
| **ELMo (Frozen Lambdas)** | 0.8675 | 0.87 | 0.87 | 0.869 |
| **CBOW** | 0.8423 | 0.85 | 0.84 | 0.844 |
| **Skip-gram** | 0.8591 | 0.86 | 0.86 | 0.859 |
| **SVD** | 0.8237 | 0.83 | 0.82 | 0.824 |

PROF

### Key Insights

- **ELMo embeddings (trainable lambdas) outperform all other methods**, demonstrating the effectiveness of contextualized embeddings.
- **CBOW and Skip-gram perform better than SVD**, as they capture word meaning more effectively in lower-dimensional spaces.
- **SVD lags behind** due to its reliance on purely statistical co-occurrence rather than contextual relationships.

## Conclusion

- **Trainable Lambdas is the best approach** for weighting ELMo layers, achieving the highest classification accuracy.

- **ELMo embeddings significantly outperform traditional word embeddings** due to their ability to capture context-dependent meaning.
- **Hyperparameter tuning plays a crucial role**, and models with adaptive weight learning perform better than static combinations.
- **Future work**: Exploring more advanced methods like attention mechanisms for embedding weighting could further improve results.

## Confusion Matrices

- `confusion_matrix_trainable_lambda.png`
- `confusion_matrix_frozen_lambda.png`
- `confusion_matrix_learnable_function.png`
- `confusion_matrix_cbows.png`
- `confusion_matrix_skipgram.png`
- `confusion_matrix_svd.png`