# REPORT - iNLP Assignment 4

## Overview

This report explores the development and performance of an ELMo (Embeddings from Language Models) architecture tailored for deep contextualised word representations, advancing from static models like word2vec and GloVe. Unlike these earlier models, ELMo generates representations that capture the nuanced meanings of words based on their contextual use within sentences, employing a bidirectional LSTM (Bi-LSTM) framework. This assignment involves constructing an ELMo model from scratch using PyTorch, pre-training it on a language modelling task, and then applying it to a 4-way classification task using the AG News Classification Dataset. We also evaluate different hyperparameter settings and compare ELMo's effectiveness against simpler word embedding models like SVD and word2vec, highlighting its impact on model performance metrics such as accuracy and F1 score.

## Downstream Task:

In this project, we utilize the ELMo architecture to tackle a 4-way classification task using the AG News Classification Dataset. This dataset includes various news articles categorized into four classes, with the task requiring the model to predict the correct category based on the text of the news description alone. Our ELMo model's performance is assessed on its ability to understand and correctly classify these texts, which involves differentiating among categories such as World, Sports, Business, and Science/Technology, thereby testing the contextual understanding capabilities of the embeddings it generates.

## Results of ELMO

## Hyperparameter Tuning:

Trainable λs :

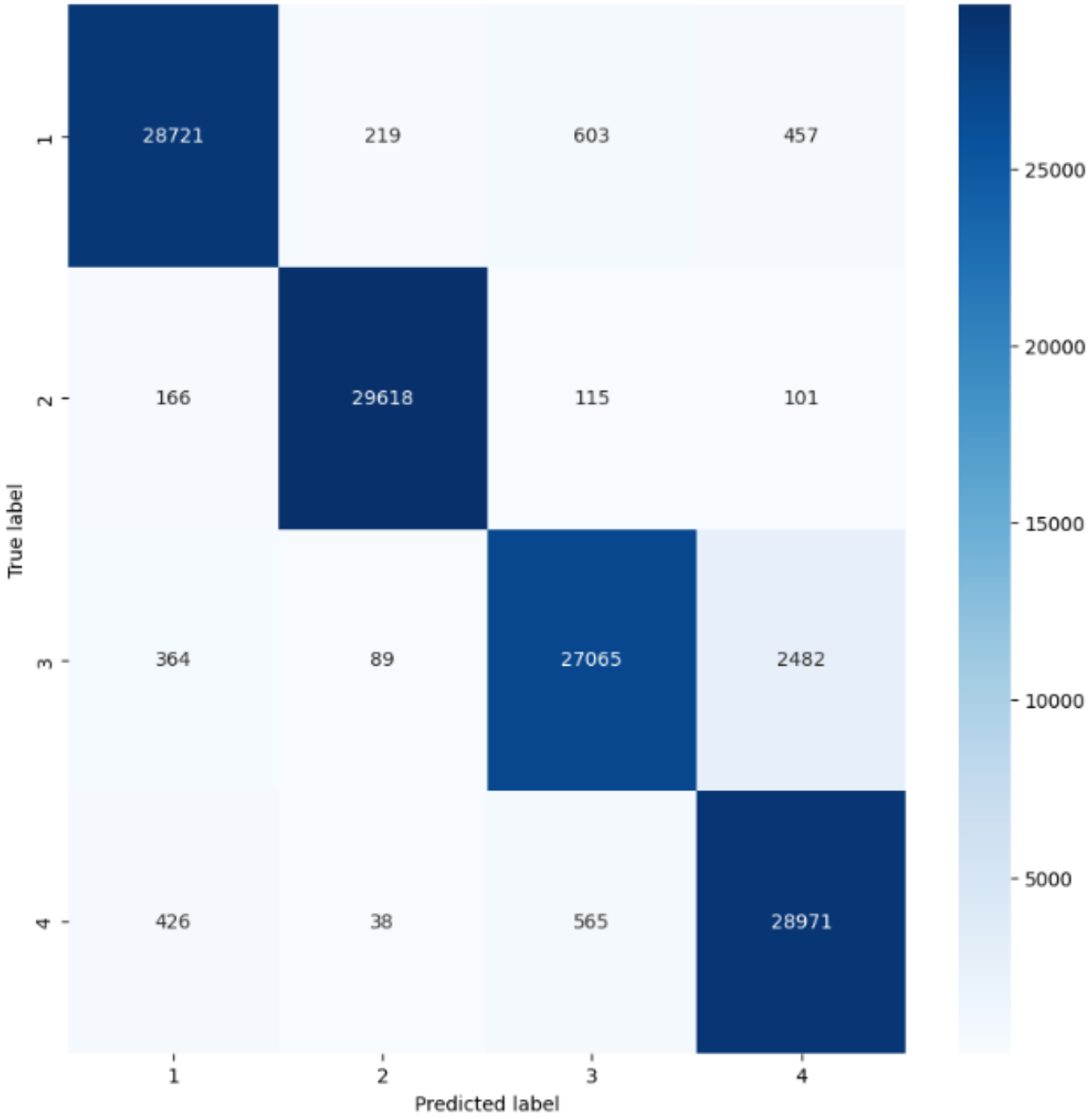
The trainable parameter gamma, denoted as λs in the ELMo model, is essential for dynamically combining the outputs of different Bi-LSTM layers. By allowing these weights to be learned, the model can adaptively determine how much emphasis to place on each layer's output during training. This is crucial because different layers capture varying levels of linguistic information—lower layers might learn more about syntax while upper layers capture contextual nuances. Hence, optimizing these weights at runtime enhances the model’s ability to generate contextually rich and precise embeddings.

**Train Loop**

```
Epoch 1/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.71it/s]
Average Loss Epoch 1: 0.37169293775955836
Epoch 2/10: 100%|██████████| 3750/3750 [02:32<00:00, 24.54it/s]
Average Loss Epoch 2: 0.2969027255485455
Epoch 3/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.77it/s]
Average Loss Epoch 3: 0.26794447795102994
Epoch 4/10: 100%|██████████| 3750/3750 [02:30<00:00, 24.94it/s]
Average Loss Epoch 4: 0.24693787955343724
Epoch 5/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.79it/s]
Average Loss Epoch 5: 0.22763379616489013
Epoch 6/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.78it/s]
Average Loss Epoch 6: 0.20978593257814646
Epoch 7/10: 100%|██████████| 3750/3750 [02:32<00:00, 24.53it/s]
Average Loss Epoch 7: 0.19370696358680725
Epoch 8/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.77it/s]
Average Loss Epoch 8: 0.17836221958721676
Epoch 9/10: 100%|██████████| 3750/3750 [02:31<00:00, 24.78it/s]
Average Loss Epoch 9: 0.1648292742483318
Epoch 10/10: 100%|██████████| 3750/3750 [02:30<00:00, 24.88it/s]
Average Loss Epoch 10: 0.15179897606571516
```

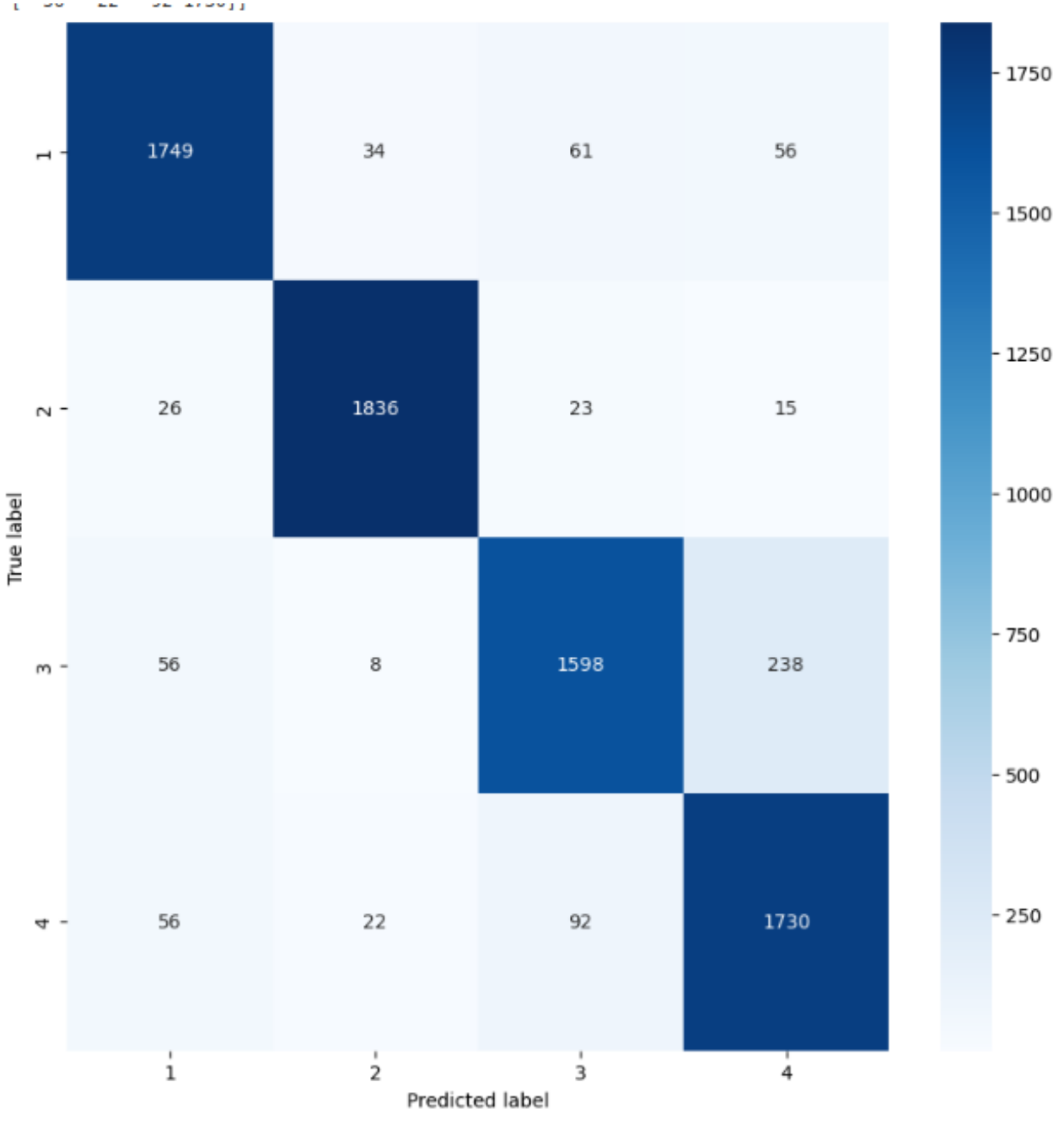
**Train Set**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.96 | 0.96 | 30000 |
| 1 | 0.99 | 0.99 | 0.99 | 30000 |
| 2 | 0.95 | 0.90 | 0.93 | 30000 |
| 3 | 0.91 | 0.97 | 0.93 | 30000 |
| accuracy | | | 0.95 | 120000 |
| macro avg | 0.95 | 0.95 | 0.95 | 120000 |
| weighted avg | 0.95 | 0.95 | 0.95 | 120000 |


28721
219
603
457
166
29618
115
101
364
89
27065
2482
426
38
565
28971
1
2
3
4
True label
Predicted label
25000
20000
15000
10000
5000

**Test Set**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.92 | 1900 |
| 1 | 0.97 | 0.97 | 0.97 | 1900 |
| 2 | 0.90 | 0.84 | 0.87 | 1900 |
| 3 | 0.85 | 0.91 | 0.88 | 1900 |
| accuracy | | | 0.91 | 7600 |
| macro avg | 0.91 | 0.91 | 0.91 | 7600 |
| weighted avg | 0.91 | 0.91 | 0.91 | 7600 |



Frozen λs:

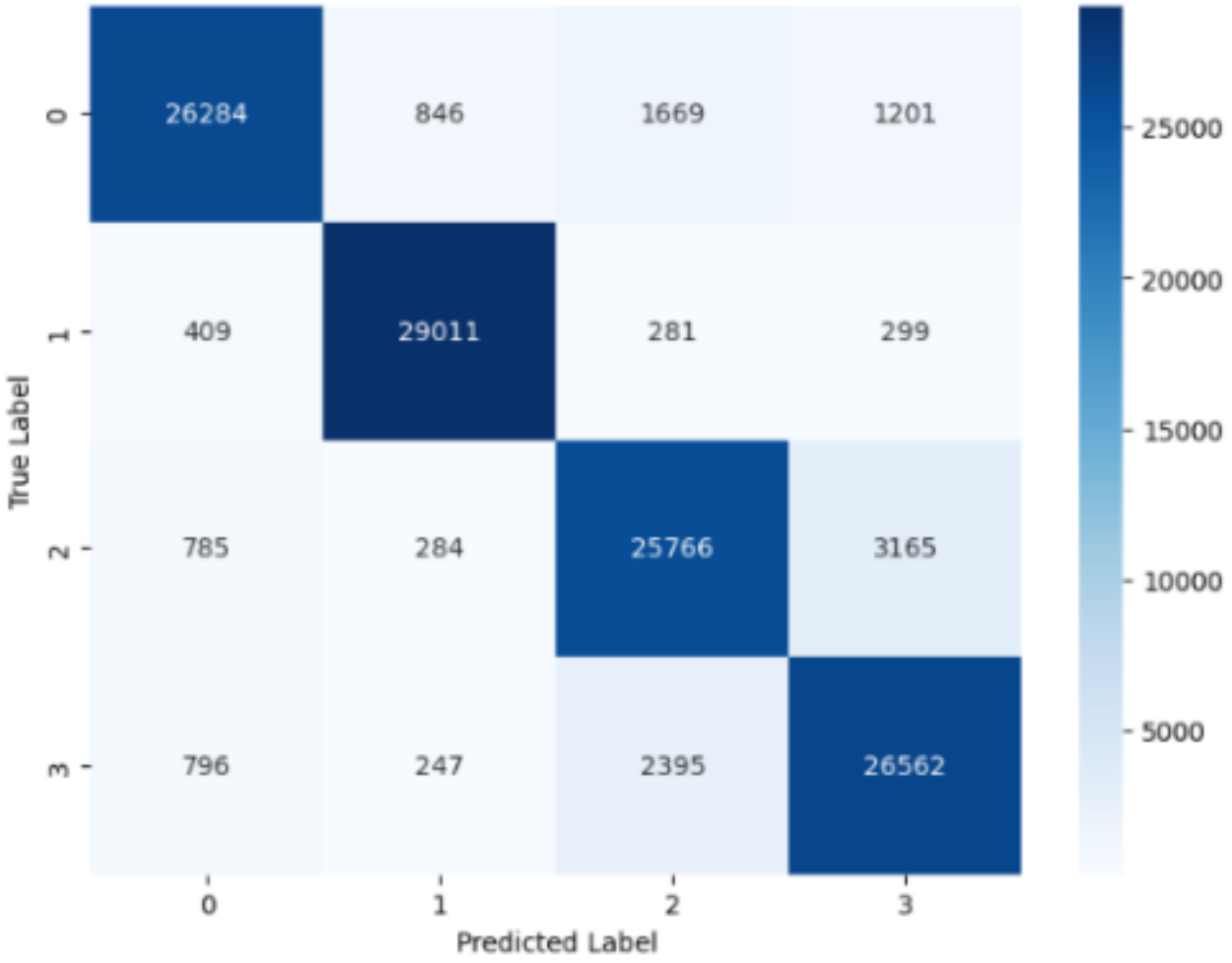
In this scenario, the λs (gamma values) are initialized to predefined constants and are kept frozen throughout the training process. This method tests the robustness of the model with static weights, assuming that each Bi-LSTM layer contributes a fixed proportion to the final word embedding. Although this approach does not allow the model to adapt λs based on the data it encounters, it simplifies the model's complexity and can provide insights into the baseline effectiveness of the predefined layer contributions. By not adjusting the weights based on training dynamics, we can evaluate the inherent capabilities of the network structure and layer configurations as initially designed.

**Train Loop**

```
Epoch 1/10: 100%|██████████| 3750/3750 [02:29<00:00, 25.08it/s]
Average Loss Epoch 1: 0.4513288195490837
Epoch 2/10: 100%|██████████| 3750/3750 [02:29<00:00, 25.06it/s]
Average Loss Epoch 2: 0.33939951030115284
Epoch 3/10: 100%|██████████| 3750/3750 [02:27<00:00, 25.34it/s]
Average Loss Epoch 3: 0.31163510419130325
Epoch 4/10: 100%|██████████| 3750/3750 [02:25<00:00, 25.72it/s]
Average Loss Epoch 4: 0.29354150732258955
Epoch 5/10: 100%|██████████| 3750/3750 [02:26<00:00, 25.68it/s]
Average Loss Epoch 5: 0.2796727279489239
Epoch 6/10: 100%|██████████| 3750/3750 [02:26<00:00, 25.57it/s]
Average Loss Epoch 6: 0.26795348560412724
Epoch 7/10: 100%|██████████| 3750/3750 [02:32<00:00, 24.61it/s]
Average Loss Epoch 7: 0.2585248796060681
Epoch 8/10: 100%|██████████| 3750/3750 [02:35<00:00, 24.12it/s]
Average Loss Epoch 8: 0.2504326263209184
Epoch 9/10: 100%|██████████| 3750/3750 [02:33<00:00, 24.47it/s]
Average Loss Epoch 9: 0.24251363901396591
Epoch 10/10: 100%|██████████| 3750/3750 [02:33<00:00, 24.50it/s]
Average Loss Epoch 10: 0.23560236271147927
```

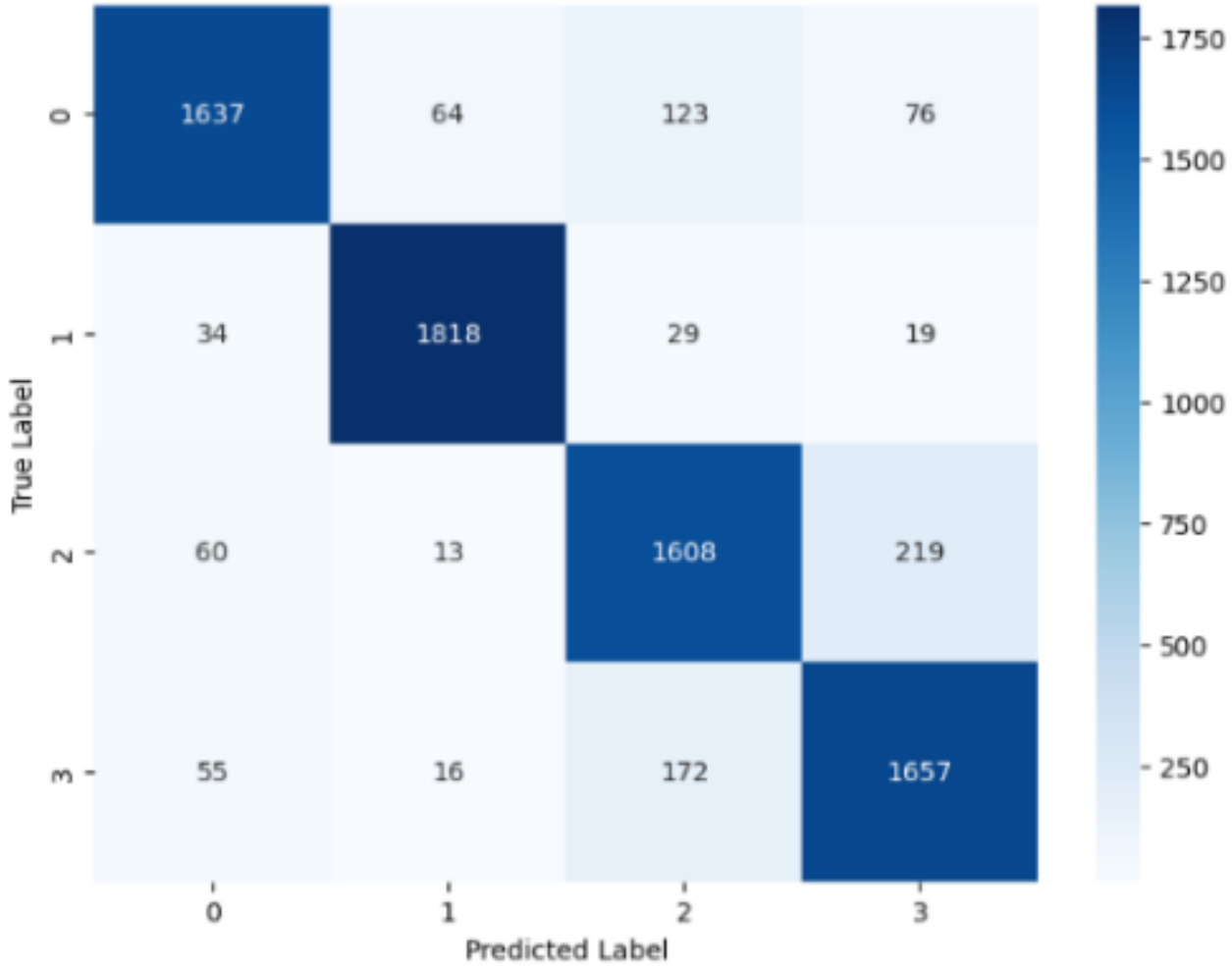
**Train Set:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.88 | 0.90 | 30000 |
| 1 | 0.95 | 0.97 | 0.96 | 30000 |
| 2 | 0.86 | 0.86 | 0.86 | 30000 |
| 3 | 0.85 | 0.89 | 0.87 | 30000 |
| accuracy | | | 0.90 | 120000 |
| macro avg | 0.90 | 0.90 | 0.90 | 120000 |
| weighted avg | 0.90 | 0.90 | 0.90 | 120000 |




**Test Set:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.86 | 0.89 | 1900 |
| 1 | 0.95 | 0.96 | 0.95 | 1900 |
| 2 | 0.83 | 0.85 | 0.84 | 1900 |
| 3 | 0.84 | 0.87 | 0.86 | 1900 |
| accuracy | | | 0.88 | 7600 |
| macro avg | 0.89 | 0.88 | 0.88 | 7600 |
| weighted avg | 0.89 | 0.88 | 0.88 | 7600 |




Learnable Function:

In this configuration, instead of using predefined or trainable scalar weights (λs), a learnable function is introduced to intelligently combine the word representations from different layers (e0, e1, e2). This function denoted as ^E = f(e0, e1,e2), is typically implemented as a neural network that learns the optimal way to merge these embeddings during training. This method allows for

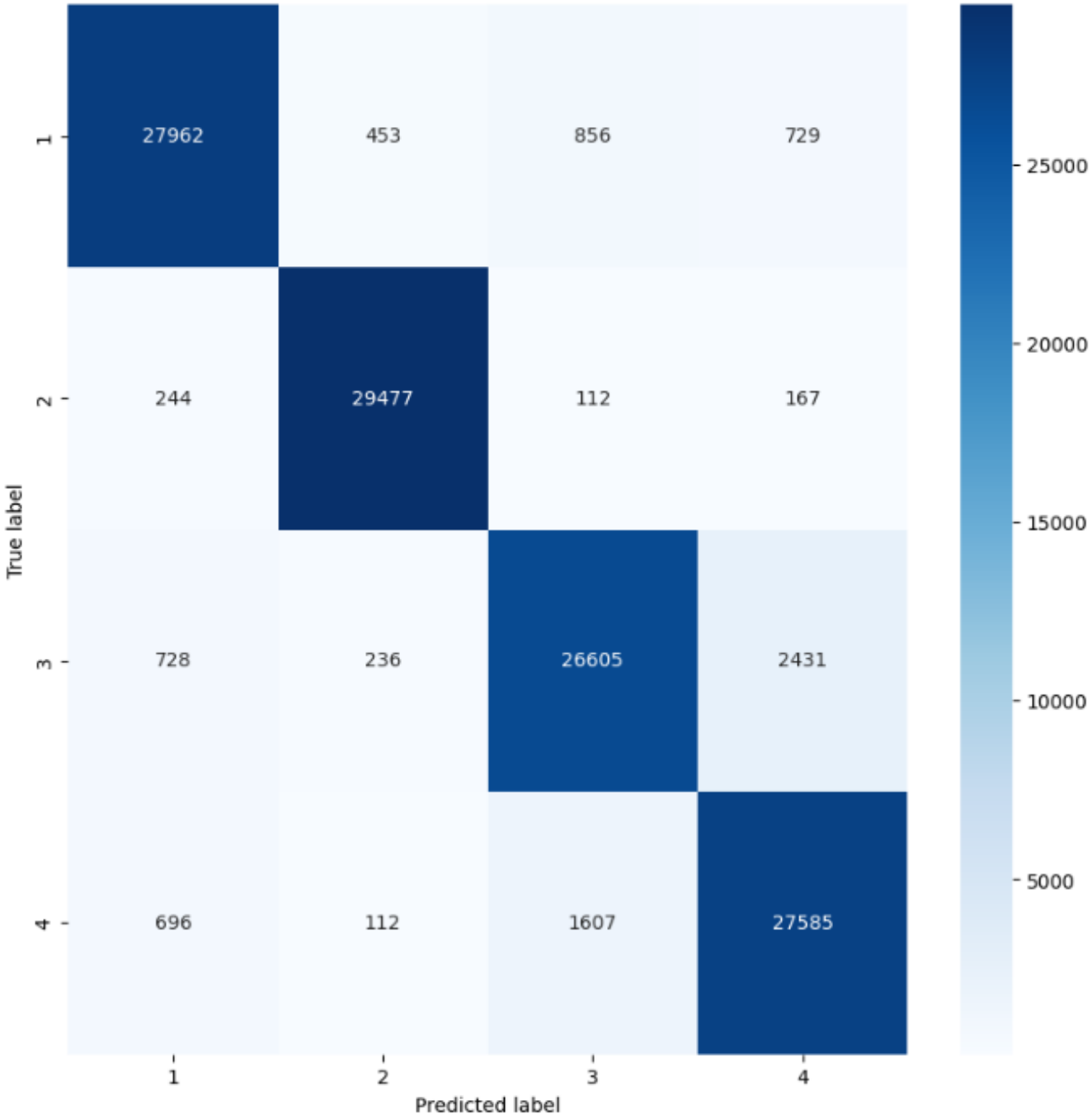
more complex interactions between the layers' outputs, potentially capturing deeper contextual relationships within the data. It can adapt to more nuanced textual patterns, potentially offering superior performance by effectively leveraging the distinct types of information encoded at eachlevel of the model's architecture.

**TRAIN LOOP**

```
Epoch 1/10: 100%|██████████| 3750/3750 [02:25<00:00, 25.80it/s]
Average Loss Epoch 1: 0.33128372422258057
Epoch 2/10: 100%|██████████| 3750/3750 [02:25<00:00, 25.84it/s]
Average Loss Epoch 2: 0.28840450771351656
Epoch 3/10: 100%|██████████| 3750/3750 [02:23<00:00, 26.06it/s]
Average Loss Epoch 3: 0.26965170167684555
Epoch 4/10: 100%|██████████| 3750/3750 [02:22<00:00, 26.25it/s]
Average Loss Epoch 4: 0.25583659464269876
Epoch 5/10: 100%|██████████| 3750/3750 [02:22<00:00, 26.26it/s]
Average Loss Epoch 5: 0.2463036407083273
Epoch 6/10: 100%|██████████| 3750/3750 [02:22<00:00, 26.28it/s]
Average Loss Epoch 6: 0.23820760096063218
Epoch 7/10: 100%|██████████| 3750/3750 [02:22<00:00, 26.25it/s]
Average Loss Epoch 7: 0.23120563428103924
Epoch 8/10: 100%|██████████| 3750/3750 [02:20<00:00, 26.77it/s]
Average Loss Epoch 8: 0.2247752978215615
Epoch 9/10: 100%|██████████| 3750/3750 [02:20<00:00, 26.69it/s]
Average Loss Epoch 9: 0.22043120498309532
Epoch 10/10: 100%|██████████| 3750/3750 [02:20<00:00, 26.64it/s]
Average Loss Epoch 10: 0.2156871337423722
```

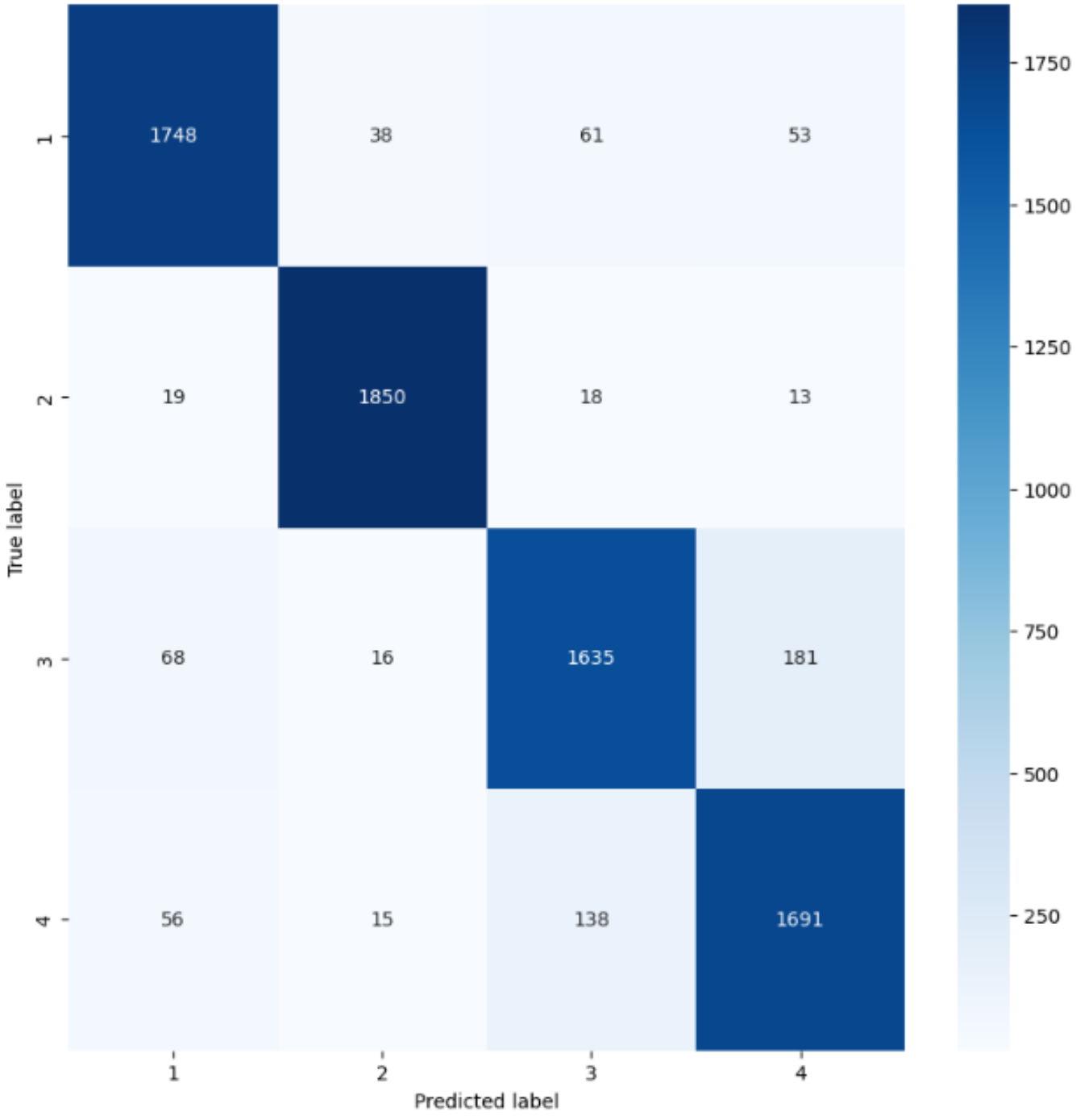
**TRAIN SET**

| | | | | |
|---|---|---|---|---|
| 0 | 0.94 | 0.93 | 0.94 | 30000 |
| 1 | 0.97 | 0.98 | 0.98 | 30000 |
| 2 | 0.91 | 0.89 | 0.90 | 30000 |
| 3 | 0.89 | 0.92 | 0.91 | 30000 |
| accuracy | | | 0.93 | 120000 |
| macro avg | 0.93 | 0.93 | 0.93 | 120000 |
| weighted avg | 0.93 | 0.93 | 0.93 | 120000 |


27962
453
856
729
244
29477
112
167
728
236
26605
2431
696
112
1607
27585
1
2
3
4
True label
Predicted label
25000
20000
15000
10000
5000

**TEST SET**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.92 | 0.92 | 1900 |
| 1 | 0.96 | 0.97 | 0.97 | 1900 |
| 2 | 0.88 | 0.86 | 0.87 | 1900 |
| 3 | 0.87 | 0.89 | 0.88 | 1900 |
| accuracy | | | 0.91 | 7600 |
| macro avg | 0.91 | 0.91 | 0.91 | 7600 |
| weighted avg | 0.91 | 0.91 | 0.91 | 7600 |



## Comparision with Skip Gram and SVD:

The Accuracy for Skip Gram on Test Set: 90 %

The Accuracy for SVD on Test Set: 81.61 %

The Accuracy for ELMO on Test Set: 91 %

This Shows that Elmo is better than SVD and Skip Gram

ELMo's superior performance over SVD and Skip Gram can be attributed to its ability to generate contextual embeddings, dynamically adjust model parameters, and leverage the power of deep learning to capture complex linguistic patterns and relationships. These factors collectively contribute to ELMo's effectiveness in tasks requiring accurate and nuanced semantic understanding of text data.