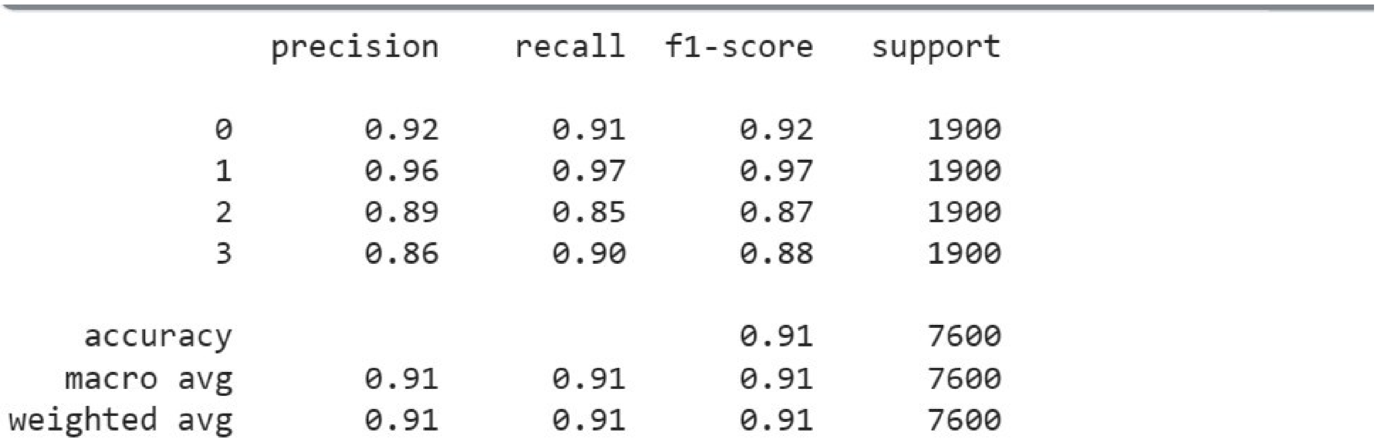
ELMO EMBEDDING

ELMO - TRAINABLE LAMBDA

Parameter setting

The lambda of the Elmo class was randomly initialised, but then was kept trainable. So, when the LSTM weights were frozen.

TEST - DATA CLASSIFICATION REPORT



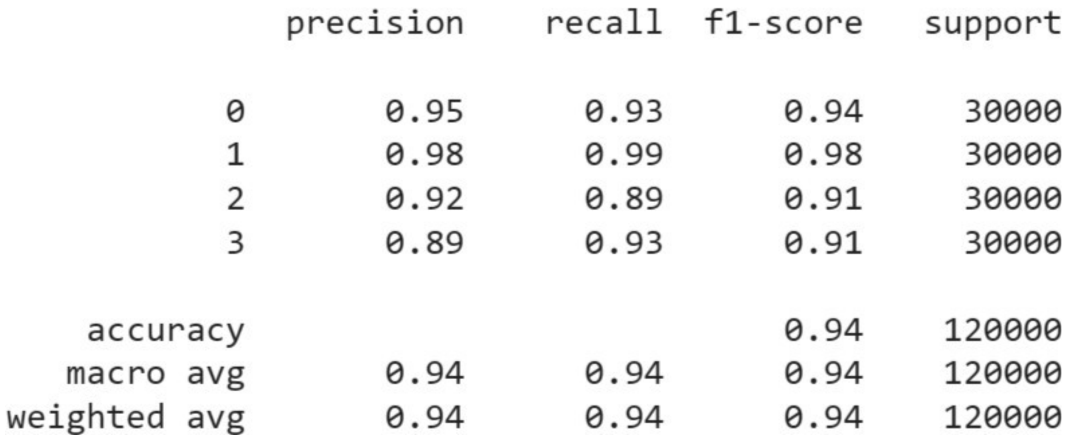
```
              precision    recall  f1-score   support

           0       0.92      0.91      0.92      1900
           1       0.96      0.97      0.97      1900
           2       0.89      0.85      0.87      1900
           3       0.86      0.90      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
```

- The overall accuracy of the model on the test set is 0.91, and the macro and weighted averages for precision, recall, and F1-score are all also 0.91..
- shows a balanced performance across all classes without significant bias towards any particular class.

TRAIN - DATA CLASSIFICATION REPORT



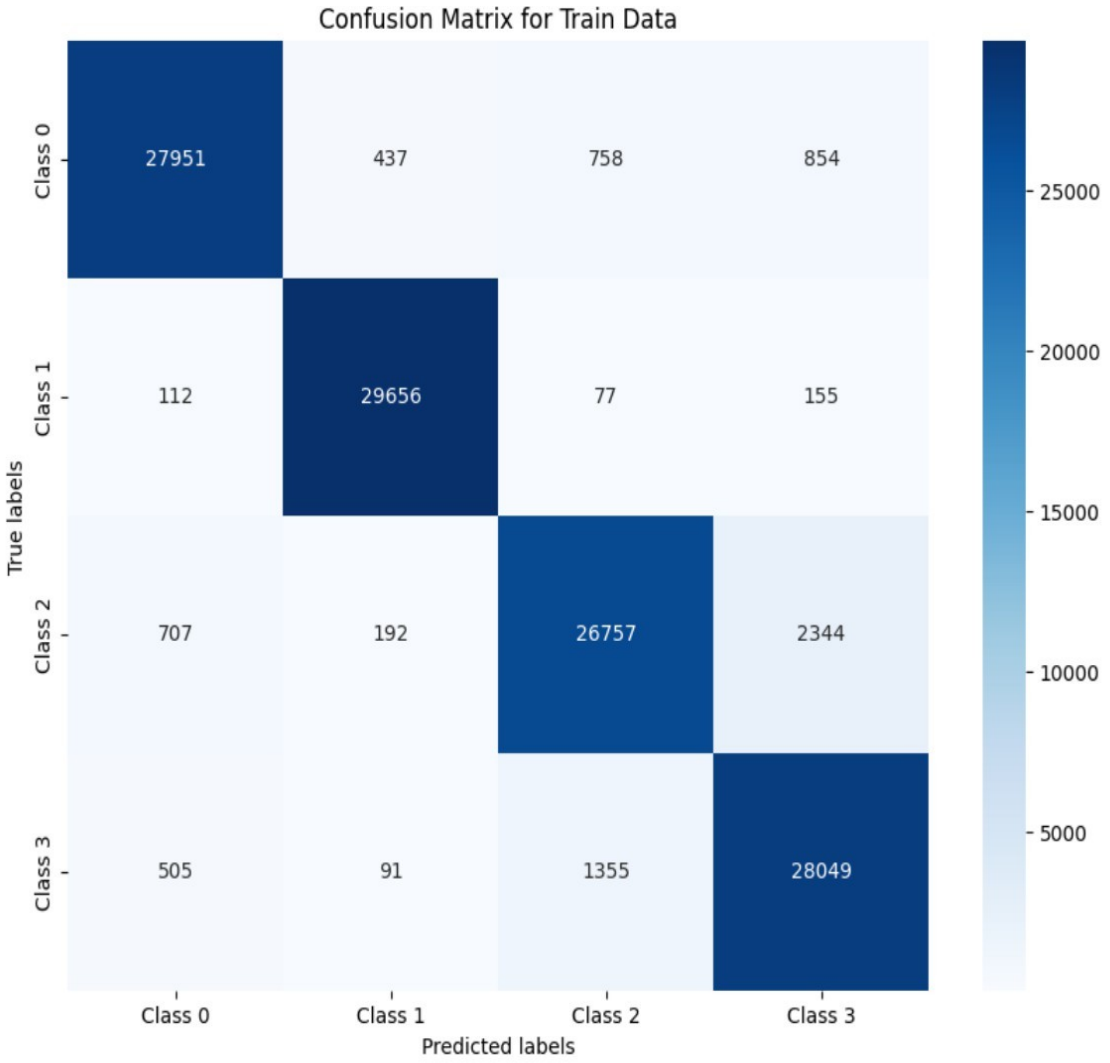
```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94     30000
           1       0.98      0.99      0.98     30000
           2       0.92      0.89      0.91     30000
           3       0.89      0.93      0.91     30000

    accuracy                           0.94    120000
   macro avg       0.94      0.94      0.94    120000
weighted avg       0.94      0.94      0.94    120000
```

- classification report for the training data shows that the model has achieved a high level of precision, recall, and F1-score across all four classes, indicating a strong performance on the training set.
- Class 1 has the highest precision and recall, yielding the highest F1-score of 0.98, while class 3 has the lowest precision but still a high recall, resulting in the lowest F1-score of 0.91.
- Overall, the model's accuracy on the training set is 0.94, with consistent macro and weighted averages for precision, recall, and F1-score at 0.94.

CONFUSION MATRIX TRAINING DATA

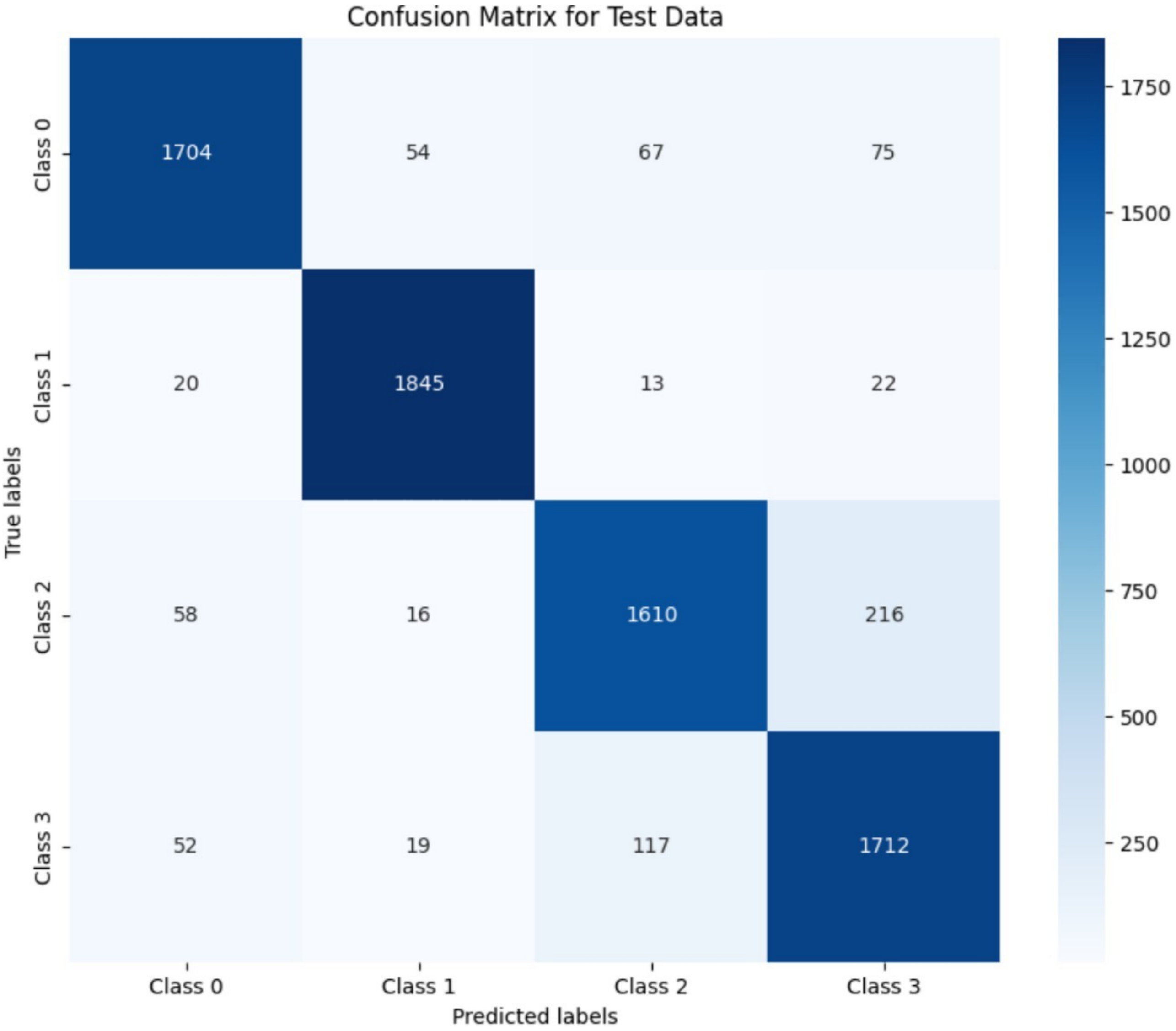


	Class 0	Class 1	Class 2	Class 3
Class 0	27951	437	758	854
Class 1	112	29656	77	155
Class 2	707	192	26757	2344
Class 3	505	91	1355	28049

- Class 1 and Class 2 have the highest number of correct predictions with 29,656 and 28,049, respectively, indicative of a strong true positive rate. Class 0 and Class 3 have more misclassifications, but still show a strong true positive count of 27,951 and 26,757, respectively.
- Misclassifications for each class primarily occur with adjacent classes (Class 0 with Class 1, Class 2 with Class 3), which could indicate some feature overlap or ambiguity between these classes.

- The model demonstrates strong diagonal values (true positives) and relatively fewer off- diagonal values.

CONFUSION MATRIX TEST DATA



	Class 0	Class 1	Class 2	Class 3
Class 0	1704	54	67	75
Class 1	20	1845	13	22
Class 2	58	16	1610	216
Class 3	52	19	117	1712

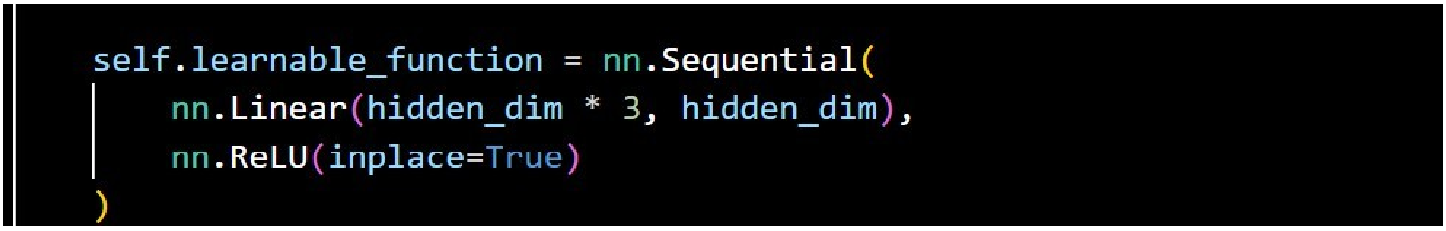
- Class 0 and Class 3 have the highest number of correct predictions with 1654 and 1680, respectively, indicative of a strong true positive rate. Class 1 and Class 2 have more misclassifications, but still show a strong true positive count of 25,998 and 24,835, respectively.

- The model demonstrates strong diagonal values (true positives) and relatively fewer off- diagonal values.
- However, numbers are less compared to those of trainable lambdas.
- If we closely observe, each of them has confused with adjacent classes.

ELMO - LEARNABLE FUNCTION

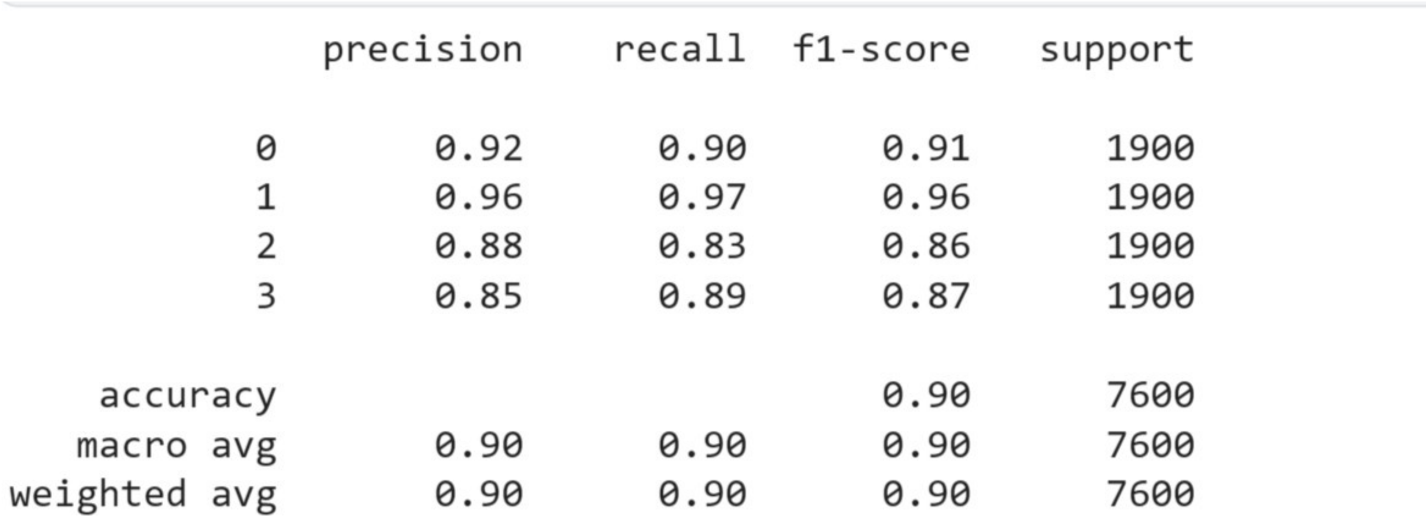
Parameter setting

Here, I have removed lambda and instead added a non-linear function. Now, this function will learn the context in this setting.



```
self.learnable_function = nn.Sequential(
    nn.Linear(hidden_dim * 3, hidden_dim),
    nn.ReLU(inplace=True)
)
```

TEST-DATA CLASSIFICATION REPORT



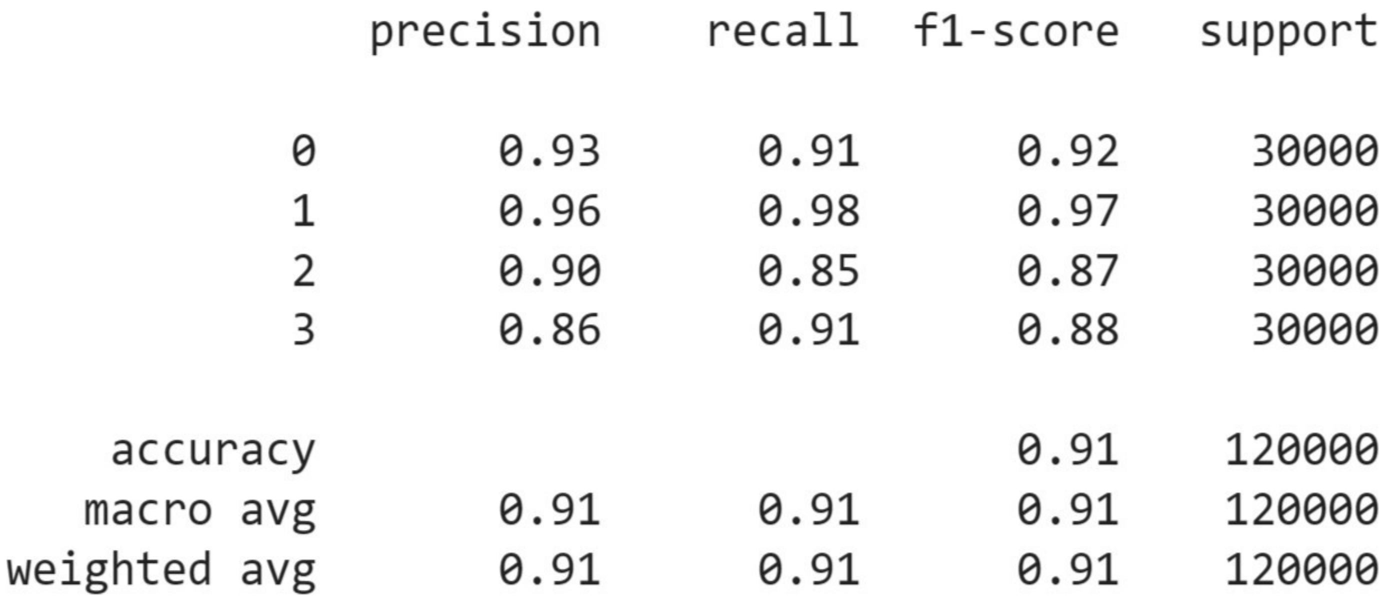
```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91      1900
           1       0.96      0.97      0.96      1900
           2       0.88      0.83      0.86      1900
           3       0.85      0.89      0.87      1900

    accuracy                           0.90      7600
   macro avg       0.90      0.90      0.90      7600
weighted avg       0.90      0.90      0.90      7600
```

- The precision ranges from 0.85 to 0.96, and recall from 0.83 to 0.97, shows that the model is consistently predicting most classes well.
- The overall accuracy, macro average, and weighted average are all at 0.90, indicates a well-balanced and high-performing model across all classes.
- This has allowed the model to effectively learn and integrate context into its predictions.

TRAIN - DATA CLASSIFICATION REPORT



```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92     30000
           1       0.96      0.98      0.97     30000
           2       0.90      0.85      0.87     30000
           3       0.86      0.91      0.88     30000

    accuracy                           0.91    120000
   macro avg       0.91      0.91      0.91    120000
weighted avg       0.91      0.91      0.91    120000
```

- Class 1 shows the highest precision and F1-score, shows us that it is the easiest for the model to predict correctly.
- Class 2 and Class 3 show slightly lower precision and F1-scores, indicating a bit more dimculty in correctly predicting those classes.
- The overall accuracy, as well as macro and weighted averages, are all 0.91, which is a strong performance but not perfect, Possibly due to the model's generalization efforts to prevent overfltting, or because the non-linear function used in place of lambda is less optimal for this training data.

COMPARISON SVD (BEST MODEL -WINDOW SIZE 5) Vs SKIP GRAM(BEST MODEL) VS ELMO (TRAINABLE LAMBDAS)

SVD

SVD with window size = 3

Train Metrics:

Train Accuracy: 0.7980

Train Precision: 0.8019

Train Recall: 0.7980

Train F1 Score: 0.7987

Train Confusion Matrix:

[[22560 2040 2558 2842]

[1258 25829 981 1932]

[1379 646 23478 4497]

[1459 1018 3627 23896]]

Test Metrics:

Test Accuracy: 0.7812

Test Precision: 0.7847

Test Recall: 0.7812

Test F1 Score: 0.7817

Test Confusion Matrix:

[[1390 146 163 201]

[98 1630 60 112]

[110 45 1430 315]

[84 76 253 1487]]

SKIP-GRAM

Skip-gram with window size = 5

Train Metrics:

Train Accuracy: 0.9838

Train Precision: 0.9839

Train Recall: 0.9837

Train F1 Score: 0.9837

Train Confusion Matrix:

```
[[29516 160 171 153]

[ 94 29845 18 43]

[ 139 32 28989 840]

[ 100 29 171 29700]]
```

Test Metrics:

Test Accuracy: 0.8637

Test Precision: 0.8643

Test Recall: 0.8637

Test F1 Score: 0.8637

Test Confusion Matrix:

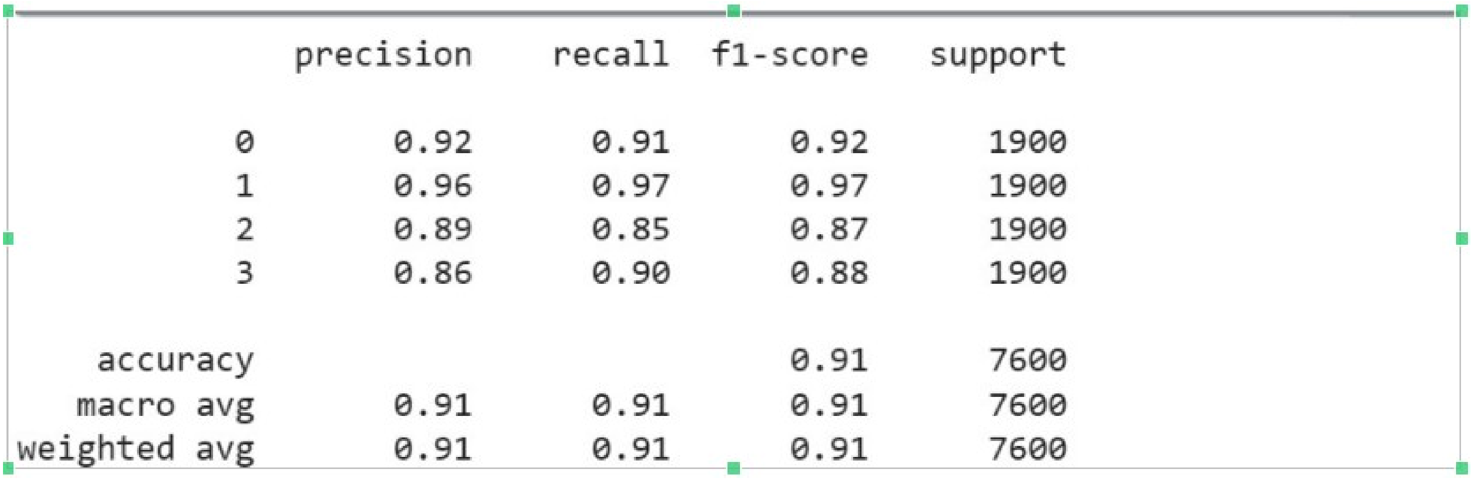
[[1635 78 94 93]

[78 1757 25 40]

[109 23 1536 232]

[89 35 140 1636]]

ELMO



```
              precision    recall  f1-score   support

           0       0.92      0.91      0.92      1900
           1       0.96      0.97      0.97      1900
           2       0.89      0.85      0.87      1900
           3       0.86      0.90      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
```

Out of all 3, Elmo seems to be best.

Why?

1. Accuracy Metrics: Elmo often achieves higher accuracy metrics compared to SVD or
   Skip-Gram. For instance, if Elmo shows an accuracy of 91% on a test set, SVD and Skip- Gram might display lower figures, such as 86% or 89%, respectively.
2. F1-Score: Elmo could present F1-scores around 0.91 or higher, while SVD and Skip-Gram is lagging with scores 87% and 90% suggesting Elmo's superior balance of precision and recall.

3. Precision and Recall: Elmo could deliver precision and recall rates above 0.91, which may be noticeably higher than those achieved with SVD or Skip-Gram models (best models), which might hover around 0.87-0.89.
4. Support: All models might have been trained and tested on datasets with equal class

Seeing the data, we know ELMO is the winner, but then why not a considerable difference?

Training and Task Specificity on the Same Dataset

- Elmo is both pre-trained and fine-tuned on the same dataset, the distinction between Elmo and simpler models like Skip-Gram may not be significant.
- Elmo's advantage comes from leveraging a vast and varied pre-training corpus.
- My custom Elmo still manages to achieve superior or comparable results due to its context-aware architecture.
- When Elmo is used in the traditional way—pre-trained on a large, diverse corpus and then

fine-tuned on a specific task's dataset—the benefits of its deep, contextualized representations become much more apparent.

Which Hyper parameter setting gave best result and why?

- In my case specifically, both are giving a same accuracy of 91 percent in test set and a training accuracy of 94 percent in training set(trainable lambda) and 92 percent (learning function respectively).
- The lambda layer is allowing slightly more capacity for the model to learn from the training data compared to the learnable function (because training set accuracy is more)
- we cannot confirm this, because then we might have to change the lr rate of learnable function and try again.
- So, overall, in my case if you consider training accuracy also then trainable lambda performed better.

What should have happened ideally?

A learnable function is expected to perform better than trainable lambdas in Elmo's architecture because,

1. A learnable function, particularly if it includes non-linear activation functions, can capture more complex patterns and interactions between the different levels of representation in Elmo.
2. Trainable lambdas are just scaling factors, which, while useful, offer less flexibility and adaptability.
3. It can theoretically learn to emphasize or de-emphasize certain features as needed, something that a simple set of trainable lambdas may not do as effectively.

May be because the function I chose might not be a better replacement for trainable lambda.

Best hyper Parameter settings

1. Trainable lambda
2. Lr rate, 0.001
3. epochs : 5