



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ  
УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ  
СІКОРСЬКОГО»  
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

INTELLECTUAL DATA ANALYSIS

Лабораторний практикум №3.

Виконав

студент гр. ФБ-51мп

Цибулено-Сігов І. М.

Перевірив

Київ 2025

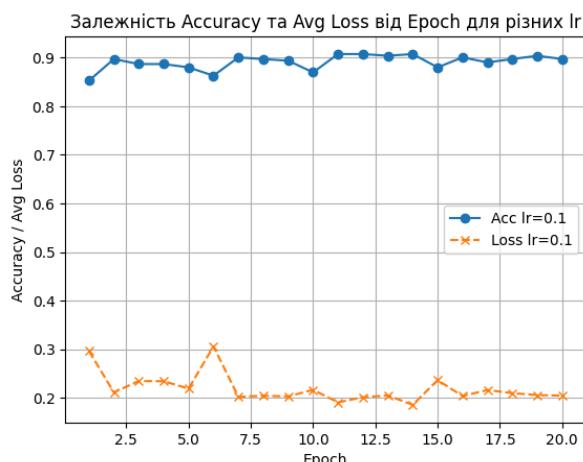
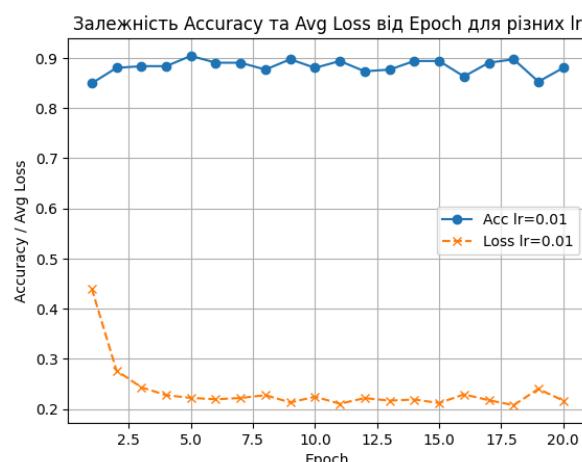
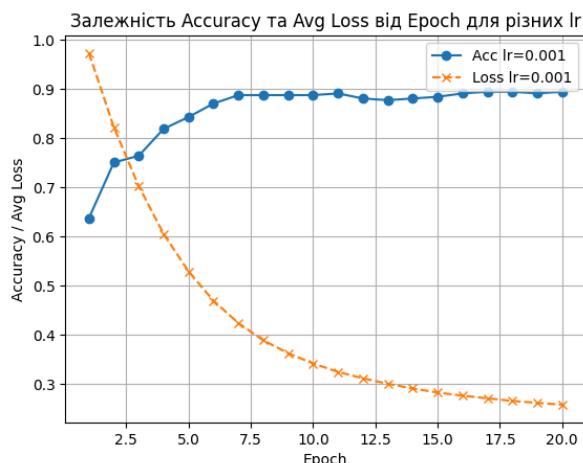
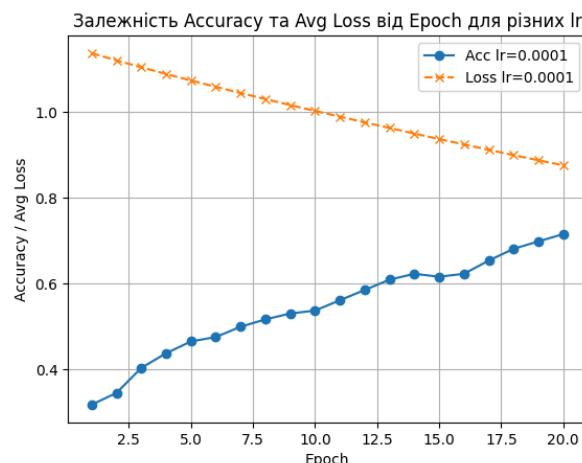
## 1. Повнозв'язані нейронні мережі

Візьміть дані, з якими ви працювали в лабораторній №1. Побудуйте повнозв'язану нейронну мережу прямого поширення для задачі класифікації. Навчіть її на тренувальній вибірці, протестуйте на тестовій. Порівняйте результати з алгоритмами з Lab 1.

Трохи пограввшись із кількістю шарів та нейронів зупинився на такій кількості:

```
FFNN(  
    (model): Sequential(  
        (0): Linear(in_features=5, out_features=16, bias=True)  
        (1): ReLU()  
        (2): Linear(in_features=16, out_features=3, bias=True)  
    )  
)
```

Також досліджу як себе веде модель при різних швидкостях навчання



Щоб зберегти стабільність залишаю lr = 0.001 а кількість епох - 20

За таких параметрів отримую результат:

Final Test Accuracy: 0.8938356164383562

В першій лабораторній найкраще справився алгоритм Decision tree

Точність моделі на тестовій вибірці: 0.9144

Точність моделі на тренувальній вибірці: 0.8987

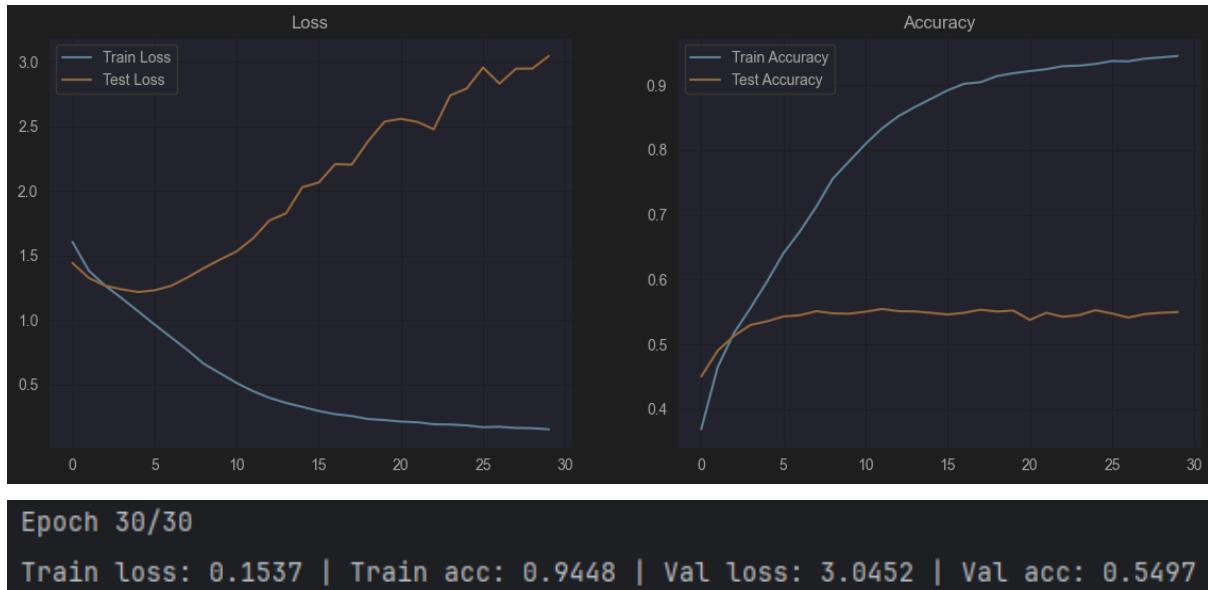
Якщо порівнювати ці моделі, то вони однаково добре прогнозують класи (різниця на рівні похибки)

## 2. Згорткові нейронні мережі

### a) Побудуйте просту згорткову нейронну мережу

(2–3 convolutional шари + fully connected). Навчіть її на обраному вами датасеті.

На цьому етапі у мене закінчилися кредити для онлайн розрахунків - пересідаю на локалхост.



З графіку явно видно перенавчання моделі. Також порівняв свій результат із результатом ентузіастів із kaggle - у них в середньому виходить 60% на 30 епосі.

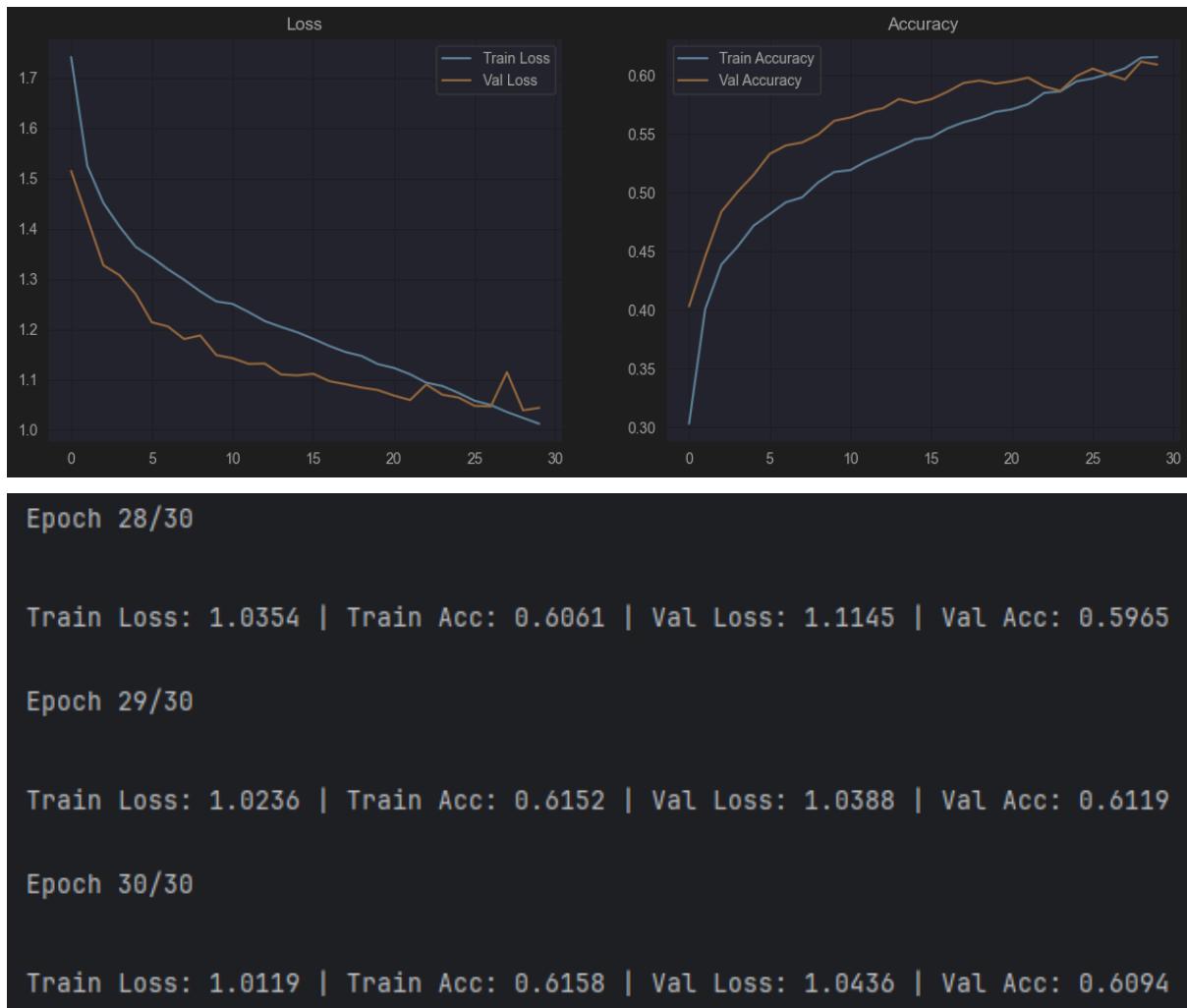
Через те, що модель тренується довго - не вистачить часу проаналізувати як саме впливає кожен параметр на модель, одразу внесу кілька змін.

Було збільшено кількість згорткових шарів з двох до трьох, що дозволяє моделі більш глибоко аналізувати локальні патерни обличчя та покращує здатність розпізнавати складні емоційні вирази. Кожен згортковий шар тепер супроводжується нормалізацією батчів (Batch Normalization), що стабілізує процес навчання та зменшує ризик перенавчання. Крім того, перед повнозв'язковими шарами доданий Dropout з ймовірністю 0.4, що додатково знижує ймовірність переобучення на тренувальних даних.

Також було впроваджено значні зміни в обробці даних. Для тренувального набору застосовані методи Data Augmentation: випадкові горизонтальні відзеркалення, обертання, а також легка зміна яскравості та контрасту. Це допомагає моделі стати більш стійкою до варіацій у зображеннях облич і покращує узагальнення на валідаційних даних. Тестовий набір залишився без змін, забезпечуючи адекватну оцінку продуктивності моделі.

Крім того, було додано автоматичне регулювання швидкості навчання через ReduceLROnPlateau, що дозволяє адаптувати learning rate при сповільненні зменшення функції втрат. Для уникнення надмірного навчання використаний механізм ранньої

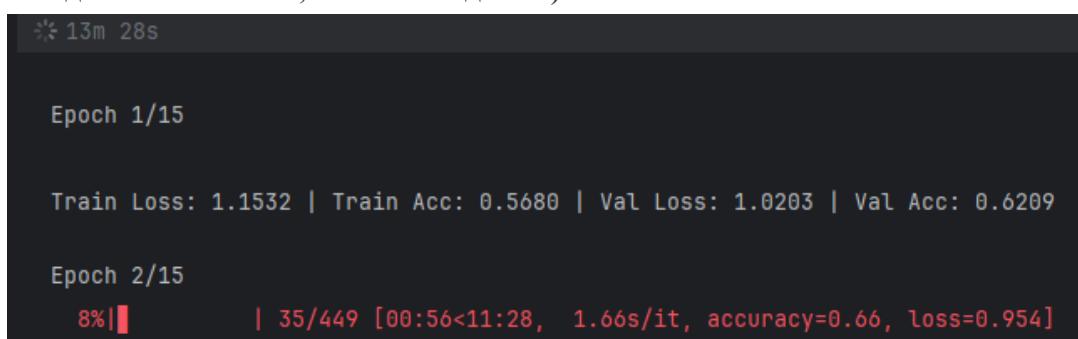
зупинки (Early Stopping), який припиняє тренування, якщо точність на валідації не покращується протягом певної кількості епох.



Тепер модель виглядає краще. з графіку видно, що якщо додати більше епох, можна отримати і кращі точність, але часу на це піде чимало.

61% точності для цього датасету вже доволі непоганий результат. На цьому зупинюємось.

**б) Використайте попередньо натреновану архітектуру (наприклад, ResNet, VGG, MobileNet). Замініть вихідний класифікатор на новий під ваші класи. Проведіть донавчання моделі на вашому датасеті. Порівняйте результати (точність, швидкість збіжності, кількість даних).**



Донавчання відбувається надзвичайно  $> 10$  хв на 1 епоху. Можливо так можна отримати кращу точність, але скоріш за все, за той же час, навчання моделі з нуля дасть кращий результат.

**3. Вирішіть задачу класифікації текстів (використайте той же датасет, з яким ви працювали в лабораторній № 2) двома способами:**

- a)** Побудуйте модель з вбудованим Embedding шаром (ініціалізованим випадковими вагами). Використайте RNN / LSTM / GRU для класифікації. Навчіть модель на вашому датасеті.
- b)** Завантажте готові embeddings (наприклад, GloVe). Ініціалізуйте Embedding шар цими вагами. Проведіть навчання.

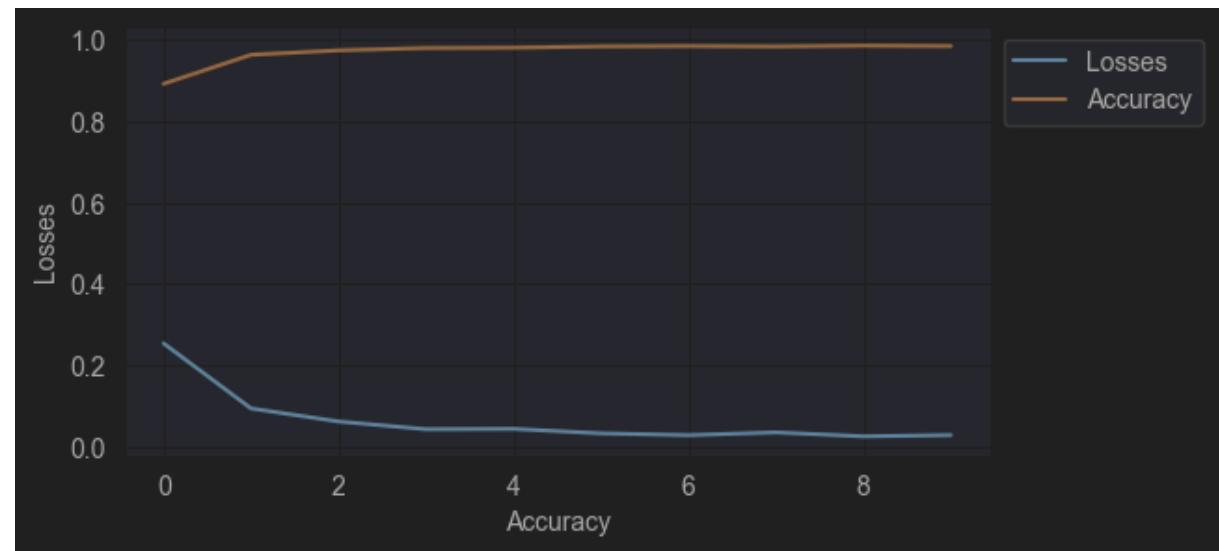
Порівняйте якість класифікації у (а) та (б). Чи покращилися метрики при використанні pretrained embeddings? Наскільки швидше/стабільніше відбулося навчання?

Я не розібрався як використати саме Pytorch для цієї задачі, тому використовуватиму іншу бібліотеку – keras.

Нейромережі не розуміють текст, вони розуміють числа. Тому перетворюю очищені тексти на послідовності індексів. (from keras.preprocessing.text import Tokenizer)

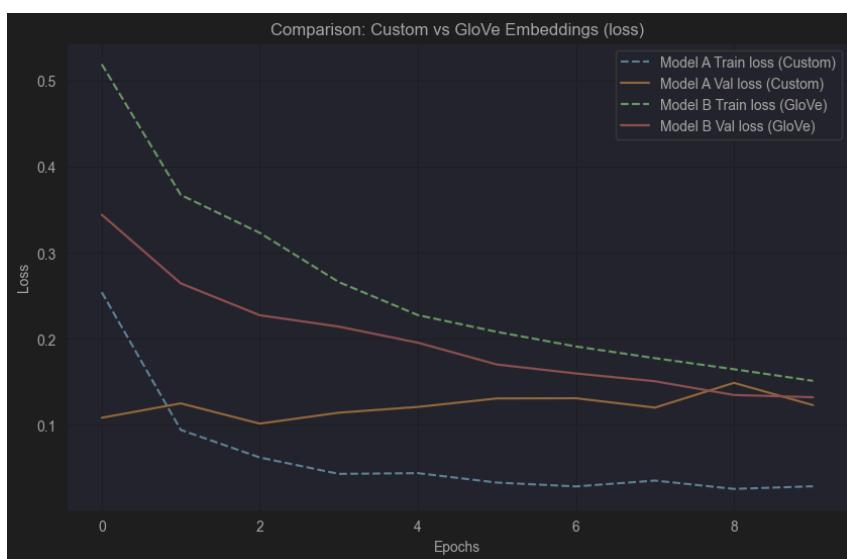
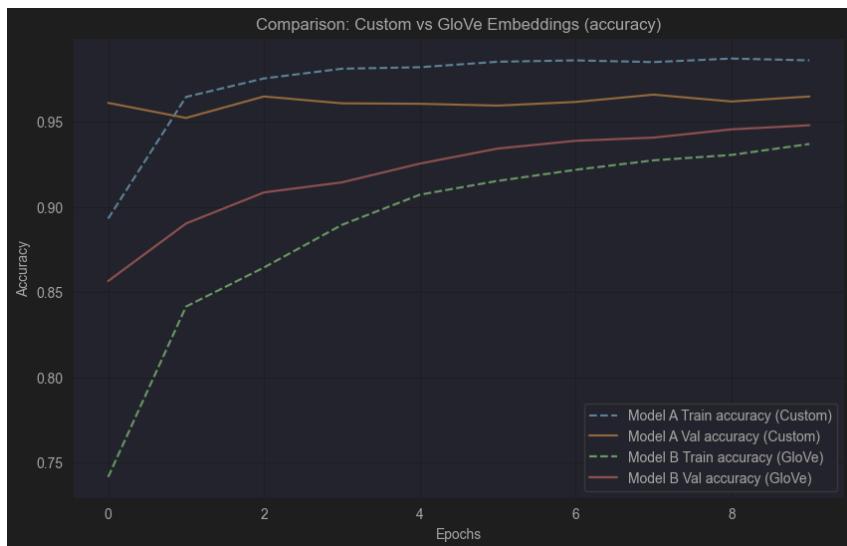
Вибрав LSTM для класифікації.

```
Epoch 8/10
233/233 9s 38ms/step - accuracy: 0.9850 - loss: 0.0352 - val_accuracy: 0.9659 - val_loss: 0.1201
Epoch 9/10
233/233 9s 38ms/step - accuracy: 0.9871 - loss: 0.0255 - val_accuracy: 0.9619 - val_loss: 0.1489
Epoch 10/10
233/233 9s 39ms/step - accuracy: 0.9860 - loss: 0.0286 - val_accuracy: 0.9649 - val_loss: 0.1230
```



Класифікація пройшла супер. Точність збігається із результатом Random Forest.

Для b) завантажую GloVe: <http://nlp.stanford.edu/data/glove.6B.zip>  
використаю набір векторів glove.6B.100d.txt



У першому випадку я використав Embedding-шар з випадковою ініціалізацією, дозволивши моделі самостійно формувати векторні представлення слів у процесі навчання. У другому випадку я ініціалізував вхідний шар готовими вагами GloVe, попередньо навченими на великих масивах текстів. Із графіків навчання можна помітити суттєву різницю: модель з власними ембедінгами швидко досягла високої точності (блізько 97%), але продемонструвала перенавчання (Loss на валідації почав зростати, а на навчання стрімко наближається до 0). Натомість модель з GloVe навчалася значно стабільніше, показуючи мінімальний розрив між тренувальними та валідаційними даними, хоча й досягла трохи нижчої пікової точності (блізько 95%).

Такий результат зумовлений лексичними особливостями датасету. Фішингові листи насичені специфічним сленгом, навмисними помилками та технічними артефактами (наприклад слова типу "urgent!!!!"), які відсутні у стандартному літературному словнику GloVe. Тому модель, що навчалася «з нуля», змогла краще адаптуватися до цих аномалій і виділити характерні ознаки спаму, тоді як універсальні ваги GloVe виявилися занадто загальними для цієї вузькоспеціалізованої задачі, хоча й забезпечили кращу узагальнюючу здатність мережі.