МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

кафедра систем штучного інтелекту



**ЗВІТ**

про виконання лабораторної роботи №1

з курсу «Проектування систем глибинного навчання»

на тему «Розробка моделей глибокого  навчання з використанням Keras»

Виконав:

*ст. групи КНСШ-12*

*Карпінський Р.М*

Перевірив:

*Пелешко Д.Д*

ЛЬВІВ – 2021

**Мета:** Виконати задані завдання за темою розбка моделей глибокого навачання з використанням Keras.

**Завдання 1**

1. Використовуючи датасет hourly\_wages\_data.csv створити модель на базі  глибокої нейронної мережі для прогнозування даних.

2. Розбити датасет на 3 підвибірки: тренувальний (70%), валідаційний (20%),  тестовий (10%), використовучі власну функцію, або функції які реалізовані в  pandas чи sklearn.

3. Дослідити як впливає скейлінг і нормалізація даних на результат моделі https://scikit-learn.org/stable/modules/preprocessing.html

4. Імплементувати модель в Keras за допомогою Sequentinal(). Обґрунтувати обрання кількості нейронів та шарів. Провести дослідження, як буде мінятися  точність мережі при різних гіперпараметрах.

5. В якості оптимізатора застосувати декілька алгоритмів навчання  (https://keras.io/api/optimizers/):

• SGD

• RMSprop

• Adam

• Adadelta

• Adagrad

• Adamax

• Nadam

6. Використати різні активаційні функції і виявити вплив виду функції та точність  моделі.

7. Побудувати графіки навчання по кожному оптимізатору і порівняти швидкість  збіжності алгоритмів.

8. Провести тестування мережі. Визначити чи не було перенавчання мережі.

**Виконання завдання 1**

train\_df = pd.read\_csv('hourly\_wages\_data.csv')

print(len(train\_df))

training\_data = train\_df.sample(frac=0.9, random\_state=25)

testing\_data = train\_df.drop(training\_data.index)

train\_X = training\_data.drop(columns=['wage\_per\_hour'])

train\_y = training\_data['wage\_per\_hour']

test\_X = testing\_data.drop(columns=['wage\_per\_hour'])

test\_y = testing\_data['wage\_per\_hour']

model = Sequential()

n\_cols = train\_X.shape[1]

model.add(Dense(10, activation='relu', input\_shape=(n\_cols,)))

model.add(Dense(10, activation='sigmoid'))

model.add(Dense(10, activation='relu'))

model.add(Dense(1))

loss = MeanSquaredError()

optim = RMSprop()

model.compile(optimizer=optim,

              loss=loss)

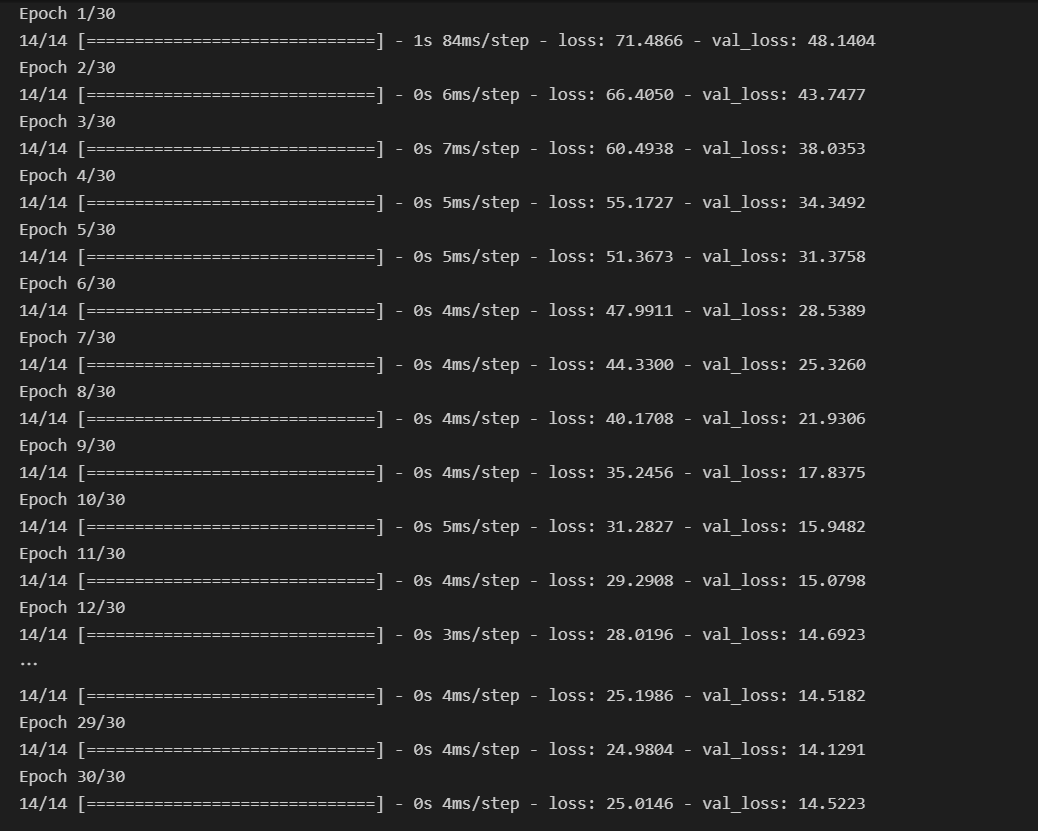
early\_stopping\_monitor = EarlyStopping(patience=8)

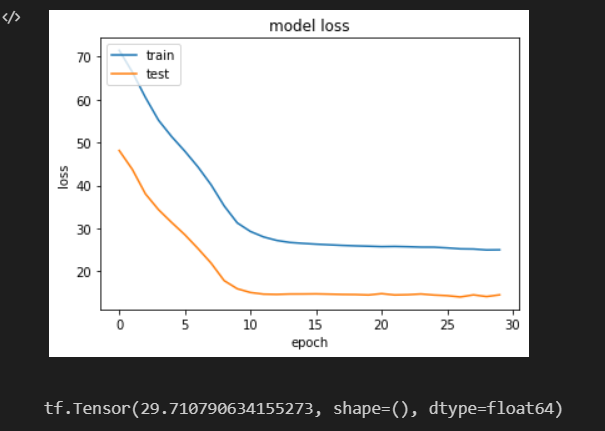
history = model.fit(train\_X, train\_y, validation\_split=0.07, epochs=30, callbacks=[early\_stopping\_monitor])

plot\_history(history)

test\_y\_preds = model.predict(test\_X)

print(loss(test\_y\_preds, test\_y))





**Завдання 2**

1. Використовуючи датасет hourly\_wages\_data.csv створити модель на базі  глибокої нейронної мережі для класифікування даних.

2. Розбити датасет на 3 підвибірки: тренувальний (70%), валідаційний (20%),  тестовий (10%), використовучі власну функцію, або функції які реалізовані в  pandas чи sklearn.

3. Дослідити як впливає скейлінг і нормалізація даних на результат моделі  https://scikit-learn.org/stable/modules/preprocessing.html

4. Імплементувати модель в Keras за допомогою Sequentinal(). Обгрунтувати  обрання кількості нейронів та шарів. Провести дослідження, як буде мінятися  точність мережі при різних гіперпараметрах.

5. В якості оптимізатора застосувати декілька алгоритмів навчання  (https://keras.io/api/optimizers/ ):

• SGD

• RMSprop

• Adam

• Adadelta

• Adagrad

• Adamax

• Nadam

6. Використати різні активаційні функції і виявити вплив виду функції та точність  моделі.

7. Побудувати графіки навчання по кожному оптимізатору і порівняти швидкість  збіжності алгоритмів.

8. Провести тестування мережі. Визначити чи не було перенавчання мережі.

**Виконання завдання 2**

train\_df = pd.read\_csv('diabetes\_data.csv')

data = np.array(train\_df)

pca = PCA(n\_components=2)

pca.fit(data)

transformed = pca.transform(data)

training\_data = train\_df.sample(frac=0.9, random\_state=25)

testing\_data = train\_df.drop(training\_data.index)

train\_X = training\_data.drop(columns=['diabetes'])

train\_y = to\_categorical(training\_data['diabetes'])

test\_X = testing\_data.drop(columns=['diabetes'])

test\_y = to\_categorical(testing\_data['diabetes'])

model = Sequential()

n\_cols = train\_X.shape[1]

model.add(Dense(10, activation='relu', input\_shape=(n\_cols,)))

model.add(Dense(10, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(10, activation='sigmoid'))

model.add(Dense(2, activation='softmax'))

loss = CategoricalCrossentropy()

optim = RMSprop()

model.compile(optimizer=optim,

loss=loss, metrics=['accuracy'])

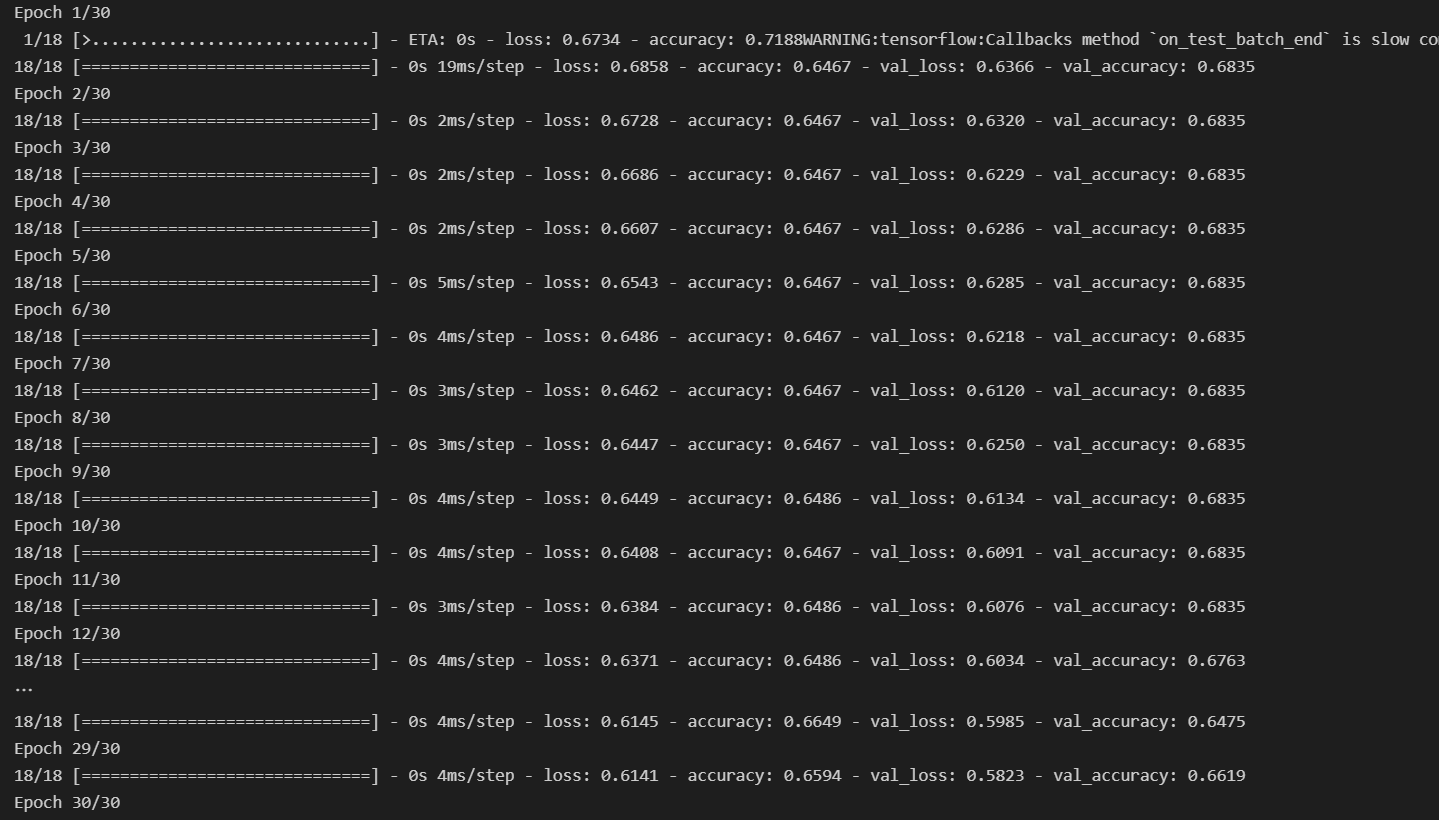
early\_stopping\_monitor = EarlyStopping(patience=9)

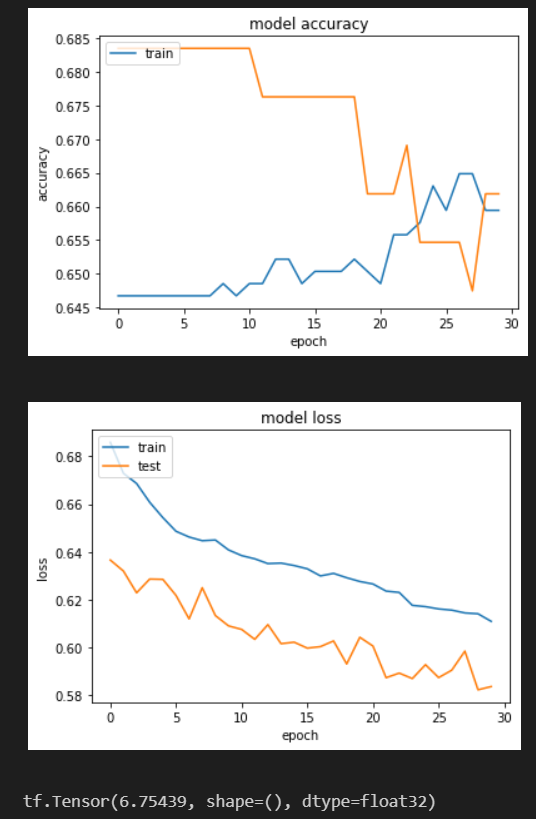
history = model.fit(train\_X, train\_y, epochs=30, validation\_split=0.2, callbacks=[])

plot\_history(history)

test\_y\_preds = model.predict(test\_X)

print(loss(test\_y\_preds, test\_y))





**Завдання 3**

1. Обрати дані з https://github.com/plotly/datasets

2. Побудувати модель глибокого навчання для задачі регресії чи задачі  класифікації (в залежності який датасет буде обрано).

3. Провести компіляцію, навчання та тестування моделі.

4. Візуалізувати результати.

**Виконання завдання 3**

train\_df = pd.read\_csv('spinrates.csv')

print(train\_df.head(10))

print(len(train\_df))

training\_data = train\_df.sample(frac=0.9, random\_state=25)

testing\_data = train\_df.drop(training\_data.index)

train\_X = training\_data.drop(columns=['velocity'])

train\_y = training\_data['velocity']

test\_X = testing\_data.drop(columns=['velocity'])

test\_y = testing\_data['velocity']

model = Sequential()

n\_cols = train\_X.shape[1]

model.add(Dense(10, activation='relu', input\_shape=(n\_cols,)))

model.add(Dense(10, activation='sigmoid'))

model.add(Dense(10, activation='relu'))

model.add(Dense(1))

loss = MeanSquaredError()

optim = RMSprop()

model.compile(optimizer=optim,

              loss=loss)

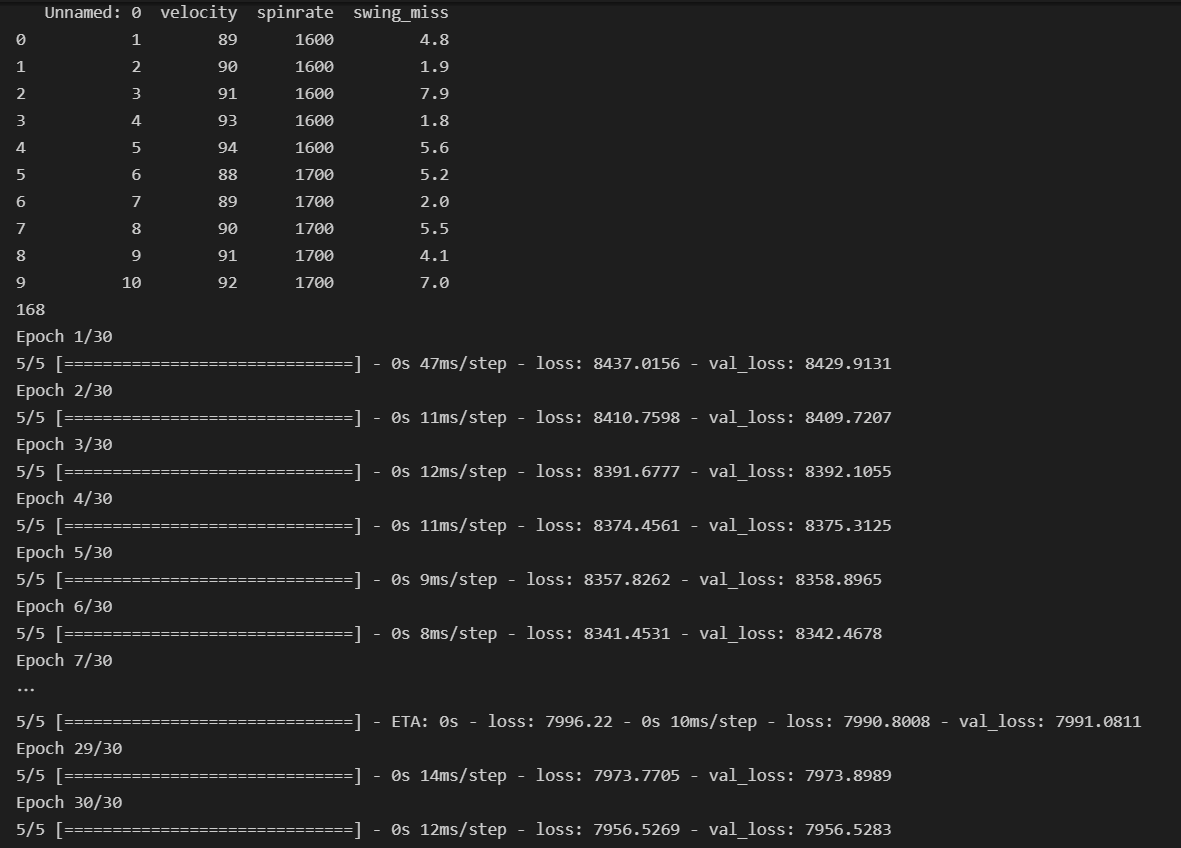
early\_stopping\_monitor = EarlyStopping(patience=3)

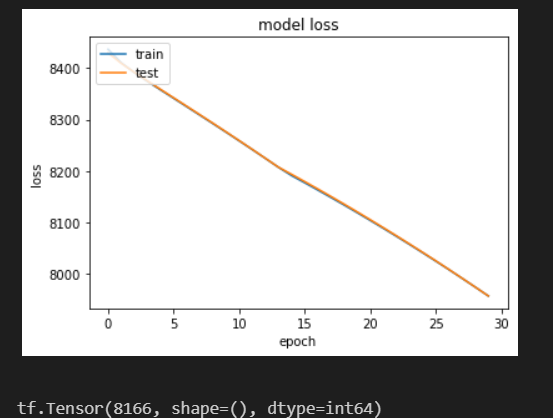
history = model.fit(train\_X, train\_y, validation\_split=0.07, epochs=30, callbacks=[early\_stopping\_monitor])

plot\_history(history)

test\_y\_preds = model.predict(test\_X)

print(loss(test\_y\_preds, test\_y))





**Висновок:** На даній лабораторній роботі, виконав поставлені завдання а саме створив модель на базі глибокої нейроної мережі, розроблення датасету, дослідження.