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.

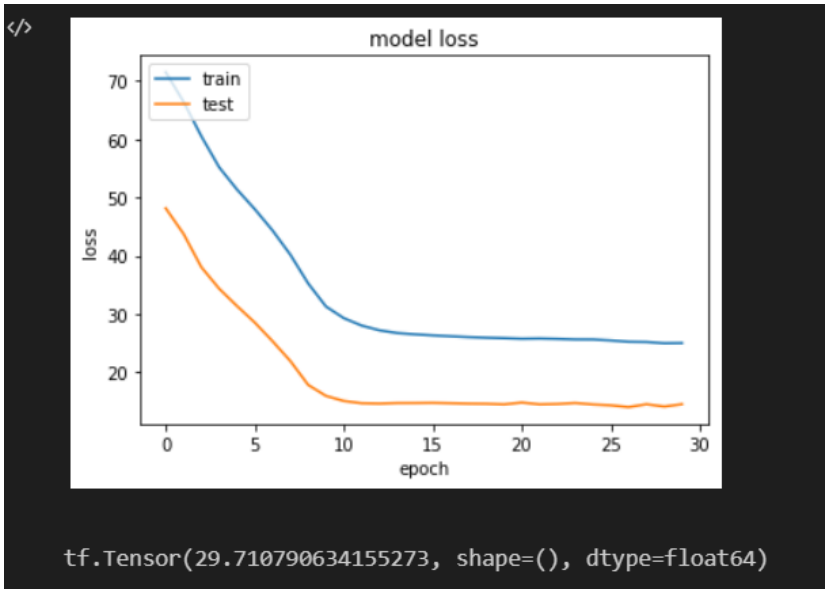
8. Провести тестування мережі. Визначити чи не було перенавчання мережі.

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))
```

```
Epoch 1/30
14/14 [==============================] - 1s 84ms/step - loss: 71.4866 - val_loss: 48.1404
Epoch 2/30
14/14 [==============================] - 0s 6ms/step - loss: 66.4050 - val_loss: 43.7477
Epoch 3/30
14/14 [==============================] - 0s 7ms/step - loss: 60.4938 - val_loss: 38.0353
Epoch 4/30
14/14 [==============================] - 0s 5ms/step - loss: 55.1727 - val_loss: 34.3492
Epoch 5/30
14/14 [==============================] - 0s 5ms/step - loss: 51.3673 - val_loss: 31.3758
Epoch 6/30
14/14 [==============================] - 0s 4ms/step - loss: 47.9911 - val_loss: 28.5389
Epoch 7/30
14/14 [==============================] - 0s 4ms/step - loss: 44.3300 - val_loss: 25.3260
Epoch 8/30
14/14 [==============================] - 0s 4ms/step - loss: 40.1708 - val_loss: 21.9306
Epoch 9/30
14/14 [==============================] - 0s 4ms/step - loss: 35.2456 - val_loss: 17.8375
Epoch 10/30
14/14 [==============================] - 0s 5ms/step - loss: 31.2827 - val_loss: 15.9482
Epoch 11/30
14/14 [==============================] - 0s 4ms/step - loss: 29.2908 - val_loss: 15.0798
Epoch 12/30
14/14 [==============================] - 0s 3ms/step - loss: 28.0196 - val_loss: 14.6923
...
14/14 [==============================] - 0s 4ms/step - loss: 25.1986 - val_loss: 14.5182
Epoch 29/30
14/14 [==============================] - 0s 4ms/step - loss: 24.9804 - val_loss: 14.1291
Epoch 30/30
14/14 [==============================] - 0s 4ms/step - loss: 25.0146 - val_loss: 14.5223
```



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.

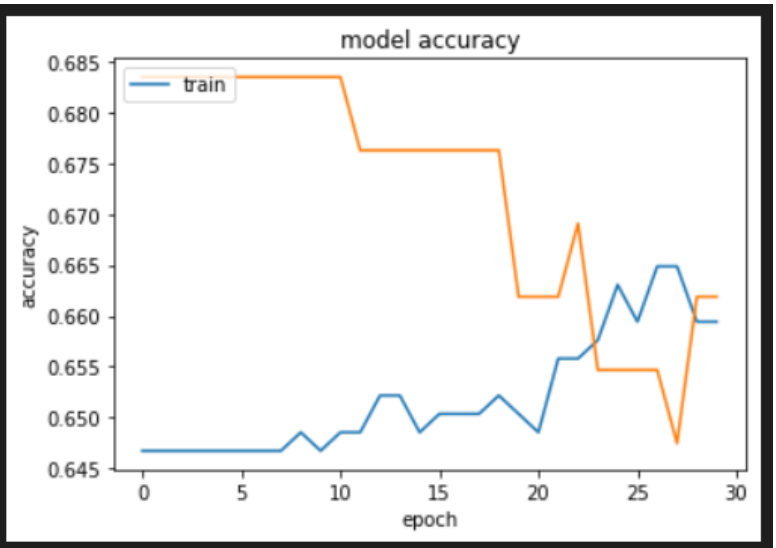
7. Побудувати графіки навчання по кожному оптимізатору і порівняти швидкість збіжності алгоритмів.

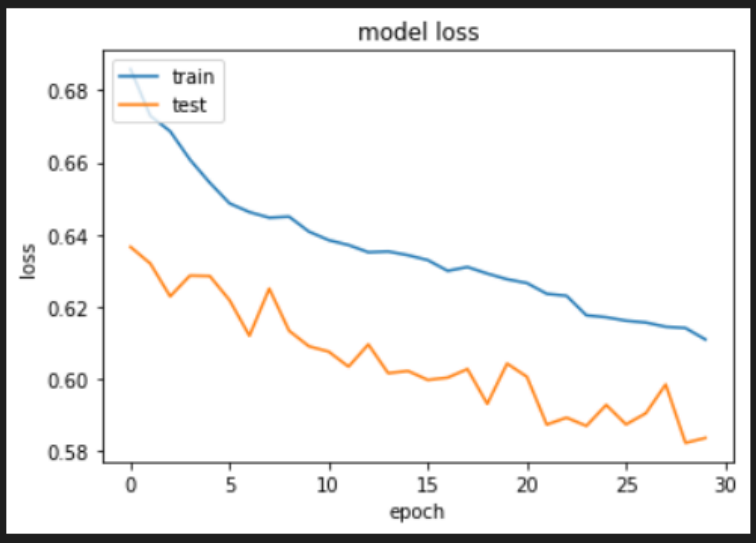
8. Провести тестування мережі. Визначити чи не було перенавчання мережі.

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))
```

```
Epoch 1/30
 1/18 [>.............................] - ETA: 0s - loss: 0.6734 - accuracy: 0.7188WARNING:tensorflow:Callbacks method `on_test_batch_end` is slow co
18/18 [==============================] - 0s 19ms/step - loss: 0.6858 - accuracy: 0.6467 - val_loss: 0.6366 - val_accuracy: 0.6835
Epoch 2/30
18/18 [==============================] - 0s 2ms/step - loss: 0.6728 - accuracy: 0.6467 - val_loss: 0.6320 - val_accuracy: 0.6835
Epoch 3/30
18/18 [==============================] - 0s 2ms/step - loss: 0.6686 - accuracy: 0.6467 - val_loss: 0.6229 - val_accuracy: 0.6835
Epoch 4/30
18/18 [==============================] - 0s 2ms/step - loss: 0.6607 - accuracy: 0.6467 - val_loss: 0.6286 - val_accuracy: 0.6835
Epoch 5/30
18/18 [==============================] - 0s 5ms/step - loss: 0.6543 - accuracy: 0.6467 - val_loss: 0.6285 - val_accuracy: 0.6835
Epoch 6/30
18/18 [==============================] - 0s 4ms/step - loss: 0.6486 - accuracy: 0.6467 - val_loss: 0.6218 - val_accuracy: 0.6835
Epoch 7/30
18/18 [==============================] - 0s 3ms/step - loss: 0.6462 - accuracy: 0.6467 - val_loss: 0.6120 - val_accuracy: 0.6835
Epoch 8/30
18/18 [==============================] - 0s 3ms/step - loss: 0.6447 - accuracy: 0.6467 - val_loss: 0.6250 - val_accuracy: 0.6835
Epoch 9/30
18/18 [==============================] - 0s 4ms/step - loss: 0.6449 - accuracy: 0.6486 - val_loss: 0.6134 - val_accuracy: 0.6835
Epoch 10/30
18/18 [==============================] - 0s 4ms/step - loss: 0.6408 - accuracy: 0.6467 - val_loss: 0.6091 - val_accuracy: 0.6835
Epoch 11/30
18/18 [==============================] - 0s 3ms/step - loss: 0.6384 - accuracy: 0.6486 - val_loss: 0.6076 - val_accuracy: 0.6835
Epoch 12/30
18/18 [==============================] - 0s 4ms/step - loss: 0.6371 - accuracy: 0.6486 - val_loss: 0.6034 - val_accuracy: 0.6763
...
18/18 [==============================] - 0s 4ms/step - loss: 0.6145 - accuracy: 0.6649 - val_loss: 0.5985 - val_accuracy: 0.6475
Epoch 29/30
18/18 [==============================] - 0s 4ms/step - loss: 0.6141 - accuracy: 0.6594 - val_loss: 0.5823 - val_accuracy: 0.6619
Epoch 30/30
```






```
tf.Tensor(6.75439, shape=(), dtype=float32)
```

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).

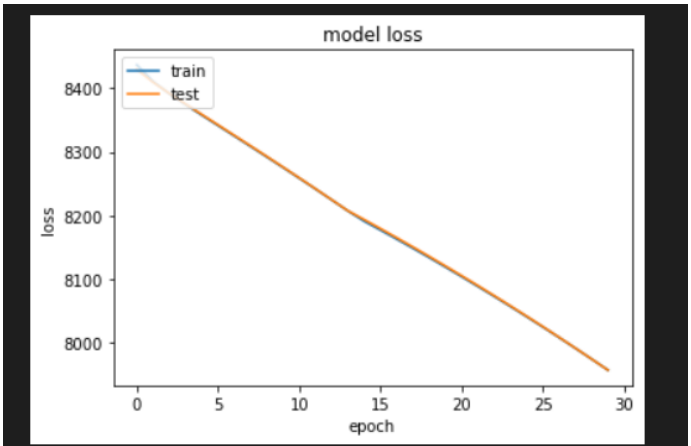
3. Провести компіляцію, навчання та тестування моделі.

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))
```

```
   Unnamed: 0  velocity  spinrate  swing_miss
0           1        89      1600         4.8
1           2        90      1600         1.9
2           3        91      1600         7.9
3           4        93      1600         1.8
4           5        94      1600         5.6
5           6        88      1700         5.2
6           7        89      1700         2.0
7           8        90      1700         5.5
8           9        91      1700         4.1
9          10        92      1700         7.0
168
Epoch 1/30
5/5 [==============================] - 0s 47ms/step - loss: 8437.0156 - val_loss: 8429.9131
Epoch 2/30
5/5 [==============================] - 0s 11ms/step - loss: 8410.7598 - val_loss: 8409.7207
Epoch 3/30
5/5 [==============================] - 0s 12ms/step - loss: 8391.6777 - val_loss: 8392.1055
Epoch 4/30
5/5 [==============================] - 0s 11ms/step - loss: 8374.4561 - val_loss: 8375.3125
Epoch 5/30
5/5 [==============================] - 0s 9ms/step - loss: 8357.8262 - val_loss: 8358.8965
Epoch 6/30
5/5 [==============================] - 0s 8ms/step - loss: 8341.4531 - val_loss: 8342.4678
Epoch 7/30
...
5/5 [==============================] - ETA: 0s - loss: 7996.22 - 0s 10ms/step - loss: 7990.8008 - val_loss: 7991.0811
Epoch 29/30
5/5 [==============================] - 0s 14ms/step - loss: 7973.7705 - val_loss: 7973.8989
Epoch 30/30
5/5 [==============================] - 0s 12ms/step - loss: 7956.5269 - val_loss: 7956.5283
```




```
tf.Tensor(8166, shape=(), dtype=int64)
```

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