

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 №2  
з курсу «Проектування систем глибинного навчання»  
на тему « Архітектура та навчання конволюційних нейронних мереж»

Виконав:

*ст. групи КНСШ-12*

*Карпінський Р.М*

Перевірив:

*Пелешко Д.Д*

**Мета:** Виконати задані завдання за темою Архітектура та навчання конволюційних нейронних мереж.

## Завдання.

1. Побудувати конволюційну мережу для розпізнавання рукописних цифр за вказаною технологією.
2. Поміняти алгоритм навчання (оптимізатор) і порівняти результати точності.
3. Поміняти розмірності ядер в конволюційних шарах – спробувати як квадратні так і прямокутні ядра.
4. Додати додаткові шари конволюції і дослідити вплив на точність розпізнавання.
5. Зробити експерименти з різними параметрам

## Виконання роботи:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings('ignore')

import os
```

```
train = pd.read_csv('train.csv')
print(train.shape)
train.head()
```

(42000, 785)

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows x 785 columns

```
test = pd.read_csv('test.csv')
print(test.shape)
test.head()
```

(28000, 784)

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows x 784 columns

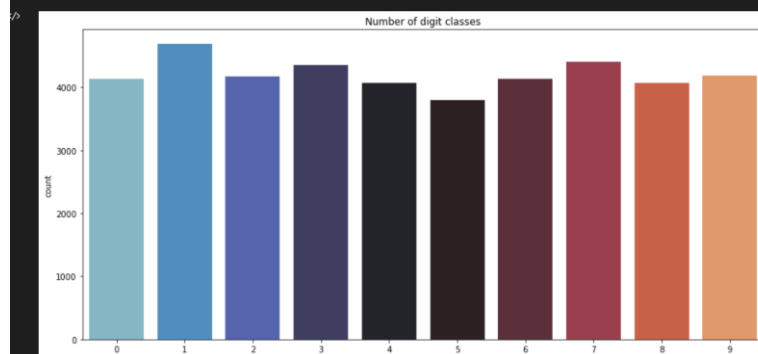
```
Y_train = train["label"]
X_train = train.drop(labels = ["label"],axis = 1)

plt.figure(figsize=(15,7))
g = sns.countplot(Y_train, palette="icefire")
plt.title("Number of digit classes")
Y_train.value_counts()
```

```

1 4684
7 4401
3 4351
9 4188
2 4177
6 4137
0 4132
4 4072
8 4063
5 3795
Name: label, dtype: int64

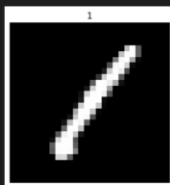
```



```

img = train.drop(labels = ["label"],axis = 1).iloc[0].to_numpy()
img = img.reshape((28,28))
plt.imshow(img,cmap='gray')
plt.title(train.iloc[0,0])
plt.axis("off")
plt.show()

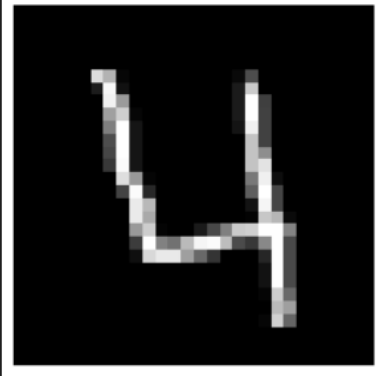
```



```

img = train.drop(labels = ["label"],axis = 1).iloc[3].to_numpy()
img = img.reshape((28,28))
plt.imshow(img,cmap='gray')
plt.title(train.iloc[3,0])
plt.axis("off")
plt.show()

```

```
...
4

X_train = X_train / 255.0
test = test / 255.0
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)

X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)

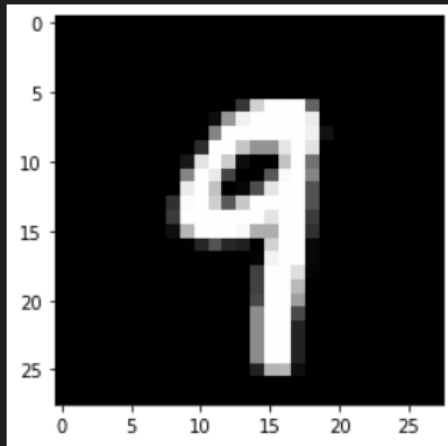
... x_train shape: (42000, 784)
test shape: (28000, 784)
x_train shape: (42000, 28, 28, 1)
test shape: (28000, 28, 28, 1)
```

```
from keras.utils.np_utils import to_categorical
Y_train = to_categorical(Y_train, num_classes = 10)

from sklearn.model_selection import train_test_split
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1
, random_state=2)
print("x_train shape",X_train.shape)
print("x_test shape",X_val.shape)
print("y_train shape",Y_train.shape)
print("y_test shape",Y_val.shape)

x_train shape (37800, 28, 28, 1)
x_test shape (4200, 28, 28, 1)
y_train shape (37800, 10)
y_test shape (4200, 10)

plt.imshow(X_train[2][:,:,0],cmap='gray')
plt.show()
```



```
import tensorflow as tf
import itertools
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D

from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau

model = Sequential()

model.add(Conv2D(filters = 8, kernel_size = (5,5),padding = 'Same',
activation = 'relu', input_shape = (28,28,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

```
model.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',
activation = 'relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))

optimizer = tf.keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)

model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])

epochs = 10
batch_size = 250
```

```

datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=0.5,
    zoom_range = 0.5,
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=False,
    vertical_flip=False
)
datagen.fit(X_train)

history = model.fit_generator(datagen.flow(X_train,Y_train, batch_size=batch_size),
epochs = epochs, validation_data = (X_val,Y_val), steps_per_epoch=X_train.shape[0] // batch_size)

```

```

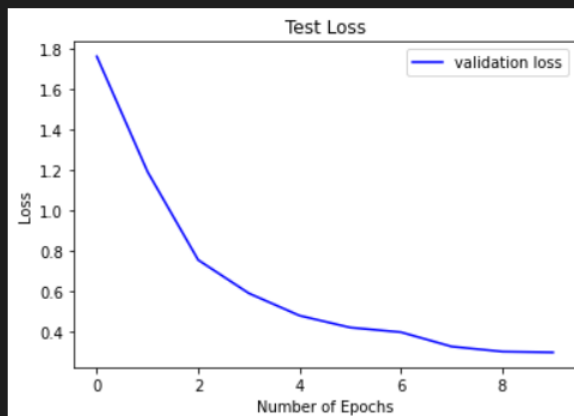
WARNING:tensorflow:From c:\Users\pavlo\AppData\Local\Temp\ipykernel_19532\3005045806.py:12: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
151/151 [=====] - 22s 142ms/step - loss: 2.1891 - accuracy: 0.1827 - val_loss: 1.7593 - val_accuracy: 0.3621
Epoch 2/10
151/151 [=====] - 20s 135ms/step - loss: 1.8941 - accuracy: 0.3244 - val_loss: 1.1912 - val_accuracy: 0.6300
Epoch 3/10
151/151 [=====] - 20s 136ms/step - loss: 1.6869 - accuracy: 0.4133 - val_loss: 0.7555 - val_accuracy: 0.8174
Epoch 4/10
151/151 [=====] - 21s 139ms/step - loss: 1.5705 - accuracy: 0.4592 - val_loss: 0.5911 - val_accuracy: 0.8633
Epoch 5/10
151/151 [=====] - 21s 138ms/step - loss: 1.4998 - accuracy: 0.4850 - val_loss: 0.4810 - val_accuracy: 0.8860
Epoch 6/10
151/151 [=====] - 21s 136ms/step - loss: 1.4504 - accuracy: 0.5031 - val_loss: 0.4222 - val_accuracy: 0.8931
Epoch 7/10
151/151 [=====] - 21s 136ms/step - loss: 1.4063 - accuracy: 0.5206 - val_loss: 0.3993 - val_accuracy: 0.8931
Epoch 8/10
151/151 [=====] - 21s 139ms/step - loss: 1.3615 - accuracy: 0.5366 - val_loss: 0.3284 - val_accuracy: 0.9148
Epoch 9/10
151/151 [=====] - 21s 139ms/step - loss: 1.3342 - accuracy: 0.5443 - val_loss: 0.3036 - val_accuracy: 0.9233
Epoch 10/10
151/151 [=====] - 21s 136ms/step - loss: 1.3048 - accuracy: 0.5562 - val_loss: 0.2998 - val_accuracy: 0.9217

```

```

plt.plot(history.history['val_loss'], color='b', label="validation loss")
plt.title("Test Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

```



```

import seaborn as sns

Y_pred = model.predict(X_val)

Y_pred_classes = np.argmax(Y_pred,axis = 1)

Y_true = np.argmax(Y_val,axis = 1)

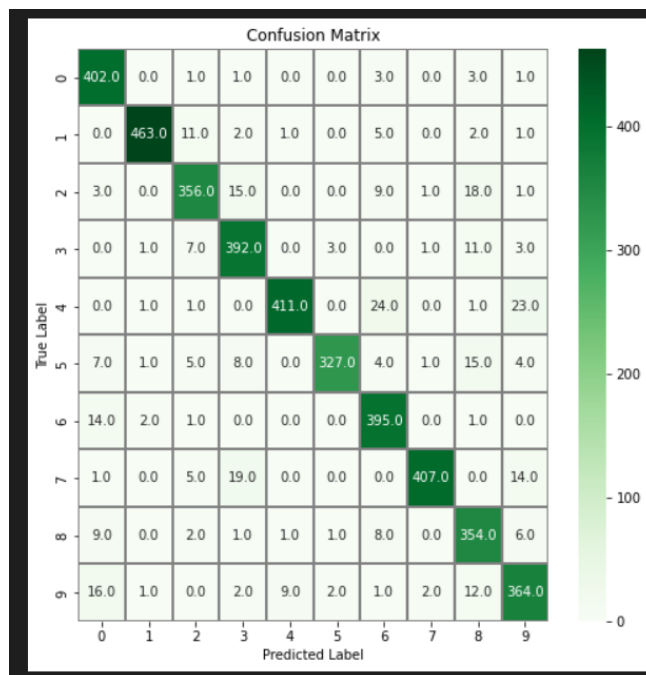
```

```

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)

f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

```



## Зробити експерименти з різними параметрам

```

model1 = Sequential()

model1.add(Conv2D(filters = 8, kernel_size = (5,5),padding = 'Same',
activation = 'relu', input_shape = (28,28,1)))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(Dropout(0.25))

# Second conv layer
model1.add(Conv2D(filters = 16, kernel_size = (4,3),padding = 'Same',
activation = 'relu'))
model1.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',
activation = 'relu'))
model1.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model1.add(Dropout(0.25))

model1.add(Flatten())
model1.add(Dense(256, activation = "relu"))
model1.add(Dropout(0.5))
model1.add(Dense(10, activation = "softmax"))
optimizer = tf.keras.optimizers.Nadam(lr=0.0001, beta_1=0.9, beta_2=0.999)
model1.compile(optimizer = optimizer, loss = "categorical_crossentropy",
metrics=["accuracy"])
datagen = ImageDataGenerator(
    featurewise_center=False,

```

```

samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=0.6,
zoom_range = 0.5,
width_shift_range=0.5,
height_shift_range=0.5,
horizontal_flip=True,
vertical_flip=True
)
datagen.fit(X_train)
Epoch 1/10
151/151 [=====] - 21s 141ms/step - loss: 1.8967 - accuracy: 0.3453 - val_loss: 0.6713 - val_accuracy: 0.8931
Epoch 2/10
151/151 [=====] - 20s 136ms/step - loss: 1.7776 - accuracy: 0.3725 - val_loss: 0.7000 - val_accuracy: 0.8681
Epoch 3/10
151/151 [=====] - 20s 134ms/step - loss: 1.7426 - accuracy: 0.3872 - val_loss: 0.7071 - val_accuracy: 0.8300
Epoch 4/10
151/151 [=====] - 21s 139ms/step - loss: 1.7090 - accuracy: 0.3981 - val_loss: 0.6891 - val_accuracy: 0.8505
Epoch 5/10
151/151 [=====] - 21s 140ms/step - loss: 1.6916 - accuracy: 0.4024 - val_loss: 0.6749 - val_accuracy: 0.8536
Epoch 6/10
151/151 [=====] - 21s 136ms/step - loss: 1.6683 - accuracy: 0.4063 - val_loss: 0.6619 - val_accuracy: 0.8657
Epoch 7/10
151/151 [=====] - 21s 136ms/step - loss: 1.6514 - accuracy: 0.4166 - val_loss: 0.6790 - val_accuracy: 0.8424
Epoch 8/10
151/151 [=====] - 21s 138ms/step - loss: 1.6463 - accuracy: 0.4186 - val_loss: 0.6720 - val_accuracy: 0.8467
Epoch 9/10
151/151 [=====] - 21s 137ms/step - loss: 1.6234 - accuracy: 0.4291 - val_loss: 0.6970 - val_accuracy: 0.7929
Epoch 10/10
151/151 [=====] - 21s 138ms/step - loss: 1.6108 - accuracy: 0.4299 - val_loss: 0.6742 - val_accuracy: 0.8186

• Rectangular conv matrix
• rotation_range=0.6
• horizontal_flip=True
• vertical_flip=True

```

```

Epoch 1/10
151/151 [=====] - 24s 159ms/step - loss: 2.3096 - accuracy: 0.1004 - val_loss: 2.2977 - val_accuracy: 0.1288
Epoch 2/10
151/151 [=====] - 22s 147ms/step - loss: 2.3001 - accuracy: 0.1119 - val_loss: 2.2941 - val_accuracy: 0.1086
Epoch 3/10
151/151 [=====] - 22s 147ms/step - loss: 2.2959 - accuracy: 0.1198 - val_loss: 2.2879 - val_accuracy: 0.1183
Epoch 4/10
151/151 [=====] - 22s 148ms/step - loss: 2.2898 - accuracy: 0.1314 - val_loss: 2.2735 - val_accuracy: 0.1855
Epoch 5/10
151/151 [=====] - 22s 146ms/step - loss: 2.2788 - accuracy: 0.1528 - val_loss: 2.2426 - val_accuracy: 0.1950
Epoch 6/10
151/151 [=====] - 22s 148ms/step - loss: 2.2557 - accuracy: 0.1670 - val_loss: 2.1773 - val_accuracy: 0.2019
Epoch 7/10
151/151 [=====] - 23s 153ms/step - loss: 2.2228 - accuracy: 0.1759 - val_loss: 2.1105 - val_accuracy: 0.2129
Epoch 8/10
151/151 [=====] - 22s 146ms/step - loss: 2.1879 - accuracy: 0.1863 - val_loss: 2.0686 - val_accuracy: 0.2174
Epoch 9/10
151/151 [=====] - 23s 149ms/step - loss: 2.1614 - accuracy: 0.1941 - val_loss: 2.0243 - val_accuracy: 0.2312
Epoch 10/10
151/151 [=====] - 22s 148ms/step - loss: 2.1430 - accuracy: 0.2025 - val_loss: 1.9751 - val_accuracy: 0.2779

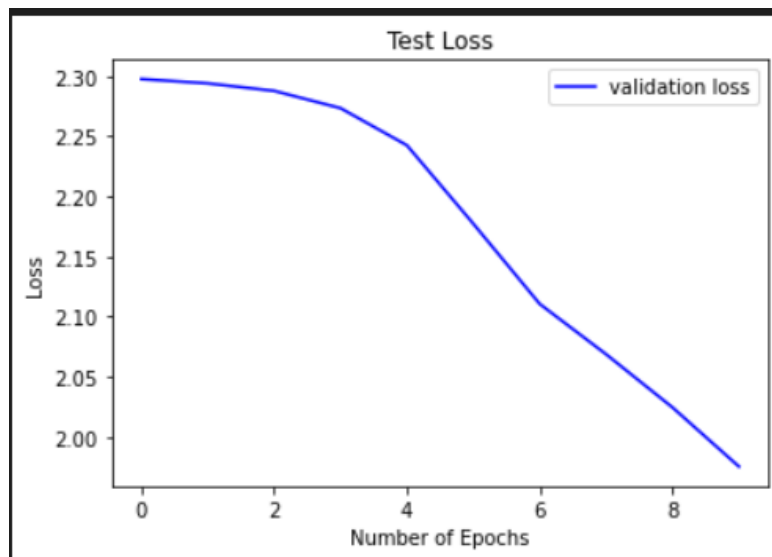
```

```

plt.plot(history.history['val_loss'], color='b', label="validation loss")
plt.title("Test Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

```





**Висновок:** На даній лабораторній роботі, виконав поставлені завдання а саме: побудував конволюційну мережу, поміняв алгоритм навчання, поміняв розмірності ядер в конволюційних шарах, додав додаткові шари конволюції, зробив експерименти з різними параметрам.