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 №4

з курсу «Проектування систем глибинного навчання»

на тему Автоенкодеритаїхтипи.Імплементація за допомогоюKeras»

Виконав:

*ст. групи КНСШ-12*

*Карпінський Р.М*

Перевірив:

*Пелешко Д.Д*

ЛЬВІВ – 2021

**Мета:** Виконати задані завдання за темою Автоенкодеритаїхтипи.Імплементація за допомогоюKeras.

**Завдання.**

* Провести дослідження двох автоенкодерів, що описані вище(глибокий автоенкодер та конволюційний автоекодер).
* Поміняти параметри моделі глибокого автоекодера длядосягненнябільшої точності і провести декілька експериментів.
* Поміняти параметри моделі конволюційного автоекодера, можливододати ще шари конволюції та пулінга (upsampling) і перебудуватиархітектуру моделі.
* Дослідити який з автоекодерів виділяє локальні особливості кращеіна яких даних.
* Спробувати по різному нанести шум на зображення і задопомогоюавтоенкодера відновити дані.

**Виконання роботи:**

from keras.layers import Dense, Input, Conv2D, LSTM, MaxPool2D, UpSampling2D

from sklearn.model\_selection import train\_test\_split

from keras.callbacks import EarlyStopping

from tensorflow.keras.utils import to\_categorical

from numpy import argmax, array\_equal

import matplotlib.pyplot as plt

from keras.models import Model

from imgaug import augmenters

from random import randint

import pandas as pd

import numpy as np

train = pd.read\_csv("fashion-mnist\_train.csv")

train\_x = train[list(train.columns)[1:]].values

train\_y = train['label'].values

## normalize and reshape the predictors

train\_x = train\_x / 255

## create train and validation datasets

train\_x, val\_x, train\_y, val\_y = train\_test\_split(train\_x, train\_y, test\_size=0.2)

## reshape the inputs

train\_x = train\_x.reshape(-1, 784)

val\_x = val\_x.reshape(-1, 784)

input\_layer = Input(shape=(784,))

## encoding architecture

encode\_layer1 = Dense(1500, activation='relu')(input\_layer)

encode\_layer2 = Dense(1000, activation='relu')(encode\_layer1)

encode\_layer3 = Dense(500, activation='relu')(encode\_layer2)

## latent view

latent\_view = Dense(10, activation='sigmoid')(encode\_layer3)

## decoding architecture

decode\_layer1 = Dense(500, activation='relu')(latent\_view)

decode\_layer2 = Dense(1000, activation='relu')(decode\_layer1)

decode\_layer3 = Dense(1500, activation='relu')(decode\_layer2)

## output layer

output\_layer = Dense(784)(decode\_layer3)

model = Model(input\_layer, output\_layer)

model.compile(optimizer='adam', loss='mse')

early\_stopping = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=10, verbose=1, mode='auto')

model.fit(train\_x, train\_x, epochs=20, batch\_size=2048, validation\_data=(val\_x, val\_x), callbacks=[early\_stopping])

*Output exceeds the [size limit](vscode-file://vscode-app/c:/Users/Akeno/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "command:workbench.action.openSettings?[\"notebook.output.textLineLimit\"]). Open the full output data[in a text editor](vscode-file://vscode-app/c:/Users/Akeno/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "command:workbench.action.openLargeOutput?18a0bbe6-9b70-4d01-b164-489d6791dd98)*

Epoch 1/20

24/24 [==============================] - 8s 330ms/step - loss: 0.0926 - val\_loss: 0.0678

Epoch 2/20

24/24 [==============================] - 7s 303ms/step - loss: 0.0637 - val\_loss: 0.0564

Epoch 3/20

24/24 [==============================] - 7s 304ms/step - loss: 0.0499 - val\_loss: 0.0435

Epoch 4/20

24/24 [==============================] - 7s 297ms/step - loss: 0.0409 - val\_loss: 0.0378

Epoch 5/20

24/24 [==============================] - 7s 297ms/step - loss: 0.0369 - val\_loss: 0.0353

Epoch 6/20

24/24 [==============================] - 7s 309ms/step - loss: 0.0333 - val\_loss: 0.0310

Epoch 7/20

24/24 [==============================] - 7s 303ms/step - loss: 0.0293 - val\_loss: 0.0275

Epoch 8/20

24/24 [==============================] - 8s 318ms/step - loss: 0.0268 - val\_loss: 0.0258

Epoch 9/20

24/24 [==============================] - 7s 300ms/step - loss: 0.0254 - val\_loss: 0.0250

Epoch 10/20

24/24 [==============================] - 8s 320ms/step - loss: 0.0242 - val\_loss: 0.0239

Epoch 11/20

24/24 [==============================] - 7s 305ms/step - loss: 0.0233 - val\_loss: 0.0229

Epoch 12/20

24/24 [==============================] - 7s 299ms/step - loss: 0.0224 - val\_loss: 0.0241

Epoch 13/20

24/24 [==============================] - 8s 315ms/step - loss: 0.0195 - val\_loss: 0.0189 Epoch 19/20 24/24 [==============================] - 7s 299ms/step - loss: 0.0186 - val\_loss: 0.0184 Epoch 20/20 24/24 [==============================] - 7s 307ms/step - loss: 0.0183 - val\_loss: 0.0181

<keras.callbacks.History at 0x21ad095dee0>

preds = model.predict(val\_x)

from PIL import Image

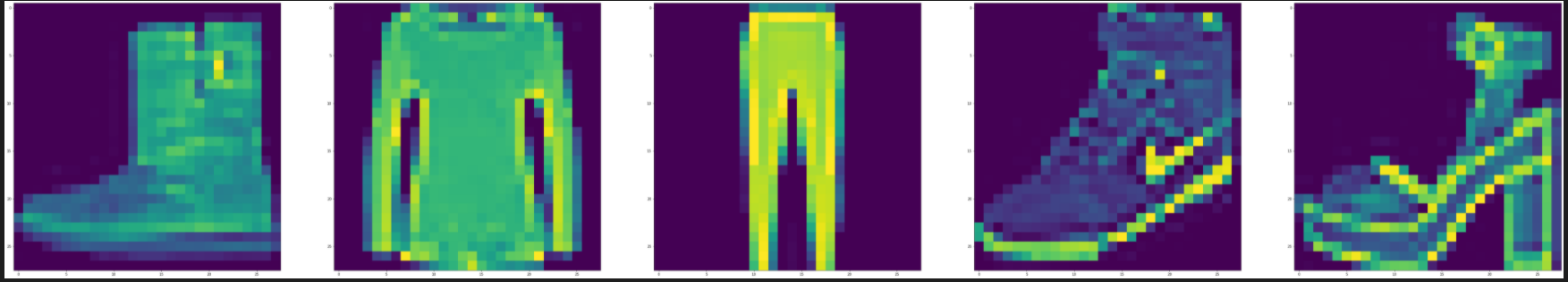
f, ax = plt.subplots(1,5)

f.set\_size\_inches(80, 40)

for i in range(5):

    ax[i].imshow(val\_x[i].reshape(28, 28))

plt.show()



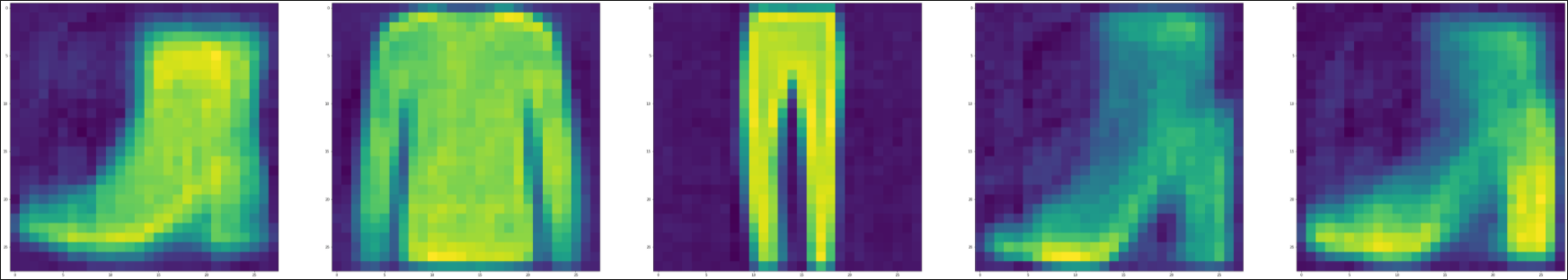
f, ax = plt.subplots(1,5)

f.set\_size\_inches(80, 40)

for i in range(5):

    ax[i].imshow(preds[i].reshape(28, 28))

plt.show()



train\_x = train[list(train.columns)[1:]].values

train\_x, val\_x = train\_test\_split(train\_x, test\_size=0.2)

## normalize and reshape

train\_x = train\_x/255.

val\_x = val\_x/255.

train\_x = train\_x.reshape(-1, 28, 28, 1)

val\_x = val\_x.reshape(-1, 28, 28, 1)

# Lets add sample noise - Salt and Pepper

noise = augmenters.SaltAndPepper(0.1)

seq\_object = augmenters.Sequential([noise])

train\_x\_n = seq\_object.augment\_images(train\_x \* 255) / 255

val\_x\_n = seq\_object.augment\_images(val\_x \* 255) / 255

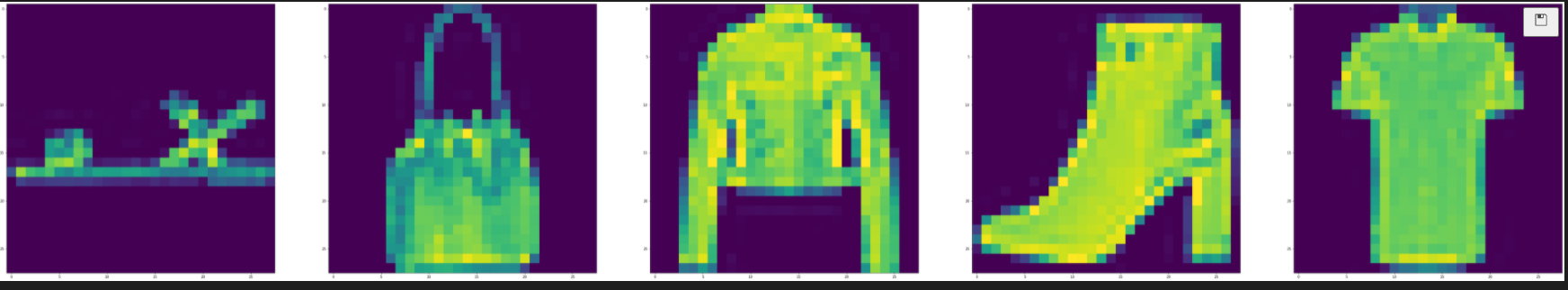
f, ax = plt.subplots(1,5)

f.set\_size\_inches(80, 40)

for i in range(5,10):

    ax[i-5].imshow(train\_x[i].reshape(28, 28))

plt.show()



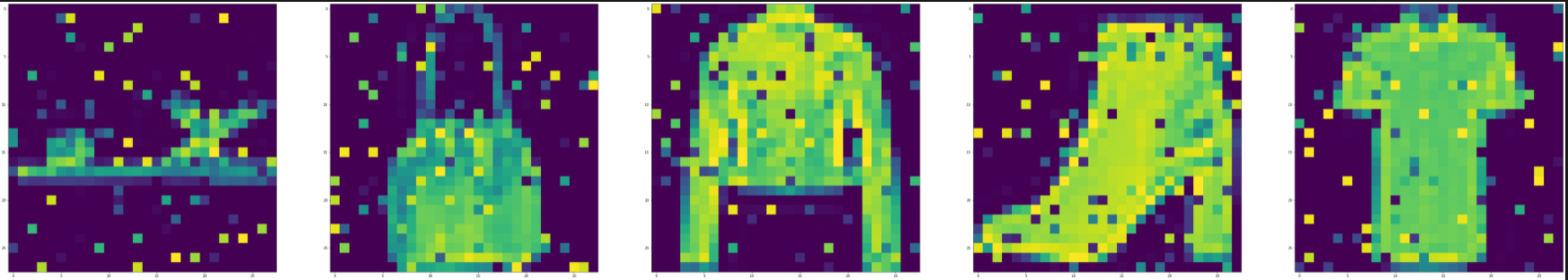
f, ax = plt.subplots(1,5)

f.set\_size\_inches(80, 40)

for i in range(5,10):

    ax[i-5].imshow(train\_x\_n[i].reshape(28, 28))

plt.show()



input\_layer = Input(shape=(28, 28, 1))

# encoding architecture

encoded\_layer1 = Conv2D(64, (3, 3), activation='relu', padding='same')(input\_layer)

encoded\_layer1 = MaxPool2D( (2, 2), padding='same')(encoded\_layer1)

encoded\_layer2 = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded\_layer1)

encoded\_layer2 = MaxPool2D( (2, 2), padding='same')(encoded\_layer2)

encoded\_layer3 = Conv2D(16, (3, 3), activation='relu', padding='same')(encoded\_layer2)

latent\_view = MaxPool2D( (2, 2), padding='same')(encoded\_layer3)

# decoding architecture

decoded\_layer1 = Conv2D(16, (3, 3), activation='relu', padding='same')(latent\_view)

decoded\_layer1 = UpSampling2D((2, 2))(decoded\_layer1)

decoded\_layer2 = Conv2D(32, (3, 3), activation='relu', padding='same')(decoded\_layer1)

decoded\_layer2 = UpSampling2D((2, 2))(decoded\_layer2)

decoded\_layer3 = Conv2D(64, (3, 3), activation='relu')(decoded\_layer2)

decoded\_layer3 = UpSampling2D((2, 2))(decoded\_layer3)

output\_layer = Conv2D(1, (3, 3), padding='same')(decoded\_layer3)

# compile the model

model\_2 = Model(input\_layer, output\_layer)

model\_2.compile(optimizer='adam', loss='mse')

model\_2.summary()

*Output exceeds the [size limit](vscode-file://vscode-app/c:/Users/Akeno/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "command:workbench.action.openSettings?[\"notebook.output.textLineLimit\"]). Open the full output data[in a text editor](vscode-file://vscode-app/c:/Users/Akeno/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "command:workbench.action.openLargeOutput?16dab4b8-15c5-4dc3-b2b4-449310294250)*

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_2 (InputLayer) [(None, 28, 28, 1)] 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d (Conv2D) (None, 28, 28, 64) 640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 14, 14, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 14, 14, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 7, 7, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 7, 7, 16) 4624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 4, 4, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 4, 4, 16) 2320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

up\_sampling2d (UpSampling2D) (None, 8, 8, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 8, 8, 32) 4640

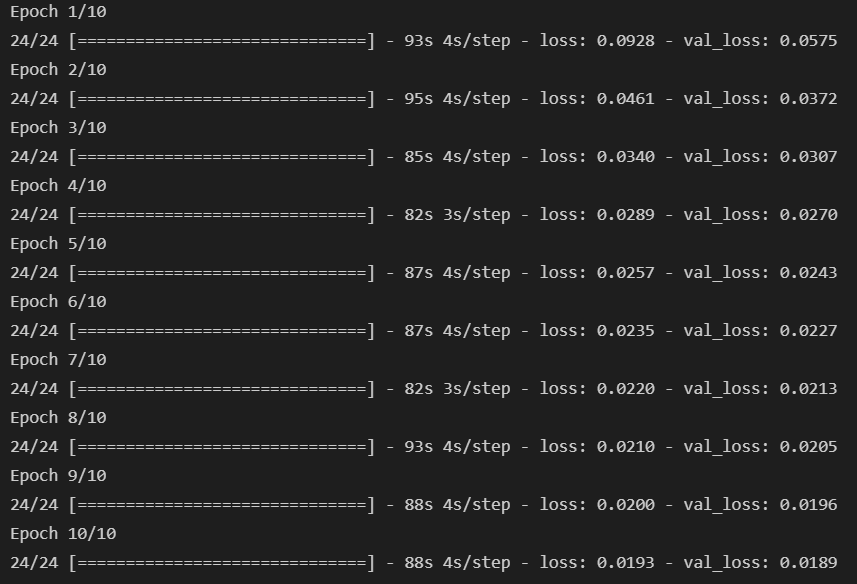
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

up\_sampling2d\_1 (UpSampling2 (None, 16, 16, 32) 0

================================================================= Total params: 49,761 Trainable params: 49,761 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

early\_stopping = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=10, verbose=5, mode='auto')

history = model\_2.fit(train\_x\_n, train\_x, epochs=10, batch\_size=2048, validation\_data=(val\_x\_n, val\_x), callbacks=[early\_stopping])



preds = model\_2.predict(val\_x\_n[:10])

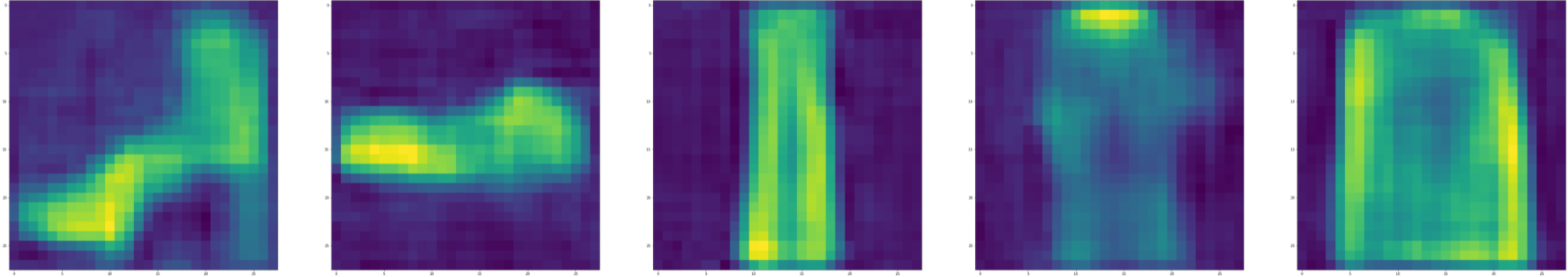
f, ax = plt.subplots(1,5)

f.set\_size\_inches(80, 40)

for i in range(5,10):

    ax[i-5].imshow(preds[i].reshape(28, 28))

plt.show()



**Висновок:** На даній лабораторній роботі, виконав поставлені завдання а саме: провів дослідження двох автоенкодерів, поміняв параметри моделі глибокого автоекодера, поміняв параметри моделі конволюційного автоекодера, дослідив який з автоекодерів виділяє локальні особливості, спробував по різному нанести шум на зображення