# Министерство науки и высшего образования Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»



## Кафедра теоретической и прикладной информатики

### Лабораторная работа №6 по дисциплине «Основы теории машинного обучения»

**Автокодировщики. Использование в целях шумоподавления**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Факультет: | ПМИ |  |  |
| Группа: | ПМИМ-01 |  |  |
| Студенты: | Ершов П.К.  Малышкина Е.Д.  Слободчикова А.Э. | | |
| Вариант: | 4 |  |  |
| Преподаватель: | Попов А.А. |  |  |
|  |  |  |  |  |

Новосибирск

2021

1. **Цель.**

Получить практические навыки по решению задачи сжатия и восстановления данных с помощью рекурентных нейронных сетей-автокодировщиков.

1. **Содержание работы.**
2. Ознакомление с теоретическими основами используемых архитектур нейронных сетей (НС).
3. Ознакомление с возможностями их реализации в рамках свободных библиотек типа TensorFlow.
4. Поиск или формирование необходимых датасетов для выбранной задачи.
5. Кодирование и отладка программы. Обучение и тестирование НС.
6. Написание отчета.
7. Защита лабораторной работы.
8. **Теоретическая часть**

Автокодировщики – особая архитектура нейронных сетей, которая позволяет применять обучение без учителя для эффективного кодирования данных. Цель автокодировщика – получить выходной сигнал наиболее приближённый к входному сигналу. Для усложнения задачи сеть делают по схеме «бутылочное горлышко» - промежуточные слои автокодировщика делаются меньше входных и выходных слоёв.

В данной работе используется рекуррентная сеть из 11 слоёв.

Для части кодировщика используются чередующиеся слои свёртки Conv2D и слои пулинга MaxPool2D.

Для декодеровщика используются чередующиеся слои свёртки Conv2D и слоёв передискретизации UpSampling2D, которые нужны для увеличения размеров данных (в данном случае, путём дублирования).

Функцией активации свёрточных слоёв выберем ReLu, так как она является хорошим аппроксиматором.

Для оптимизации выберем АdaDelta, так как она лишена недостатков АdaGrad – накопления квадратов градиентов, и тем самым предотвращает остановку процесса обучения.

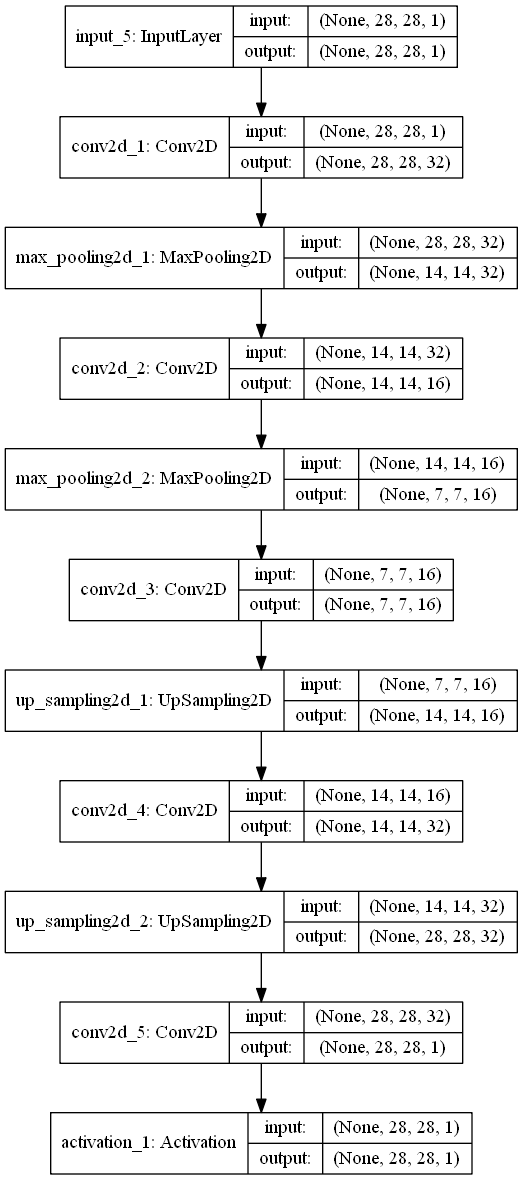


Рисунок 1. Структура нейронной сети

Обучение будет проводиться с помощью MNIST - набора данных, представляющих рукописные цифровые изображения. Для усложнения работы сети входные изображения будут зашумляться Гауссовским распределеним.

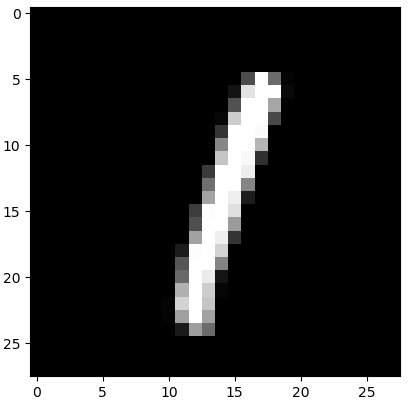
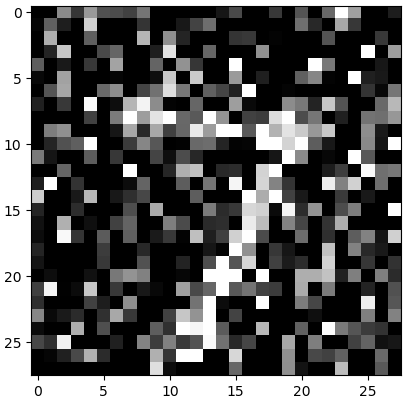
 

Рисунок 2. Исходные данные (слева) и данные с шумом (справа)

Число эпох 20. Число батчей в эпохах 64.

1. **Ход работы**

Первая архитектура:

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

Layer (type) Output Shape Param #

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

input\_5 (InputLayer) (None, 28, 28, 1) 0

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

conv2d\_1 (Conv2D) (None, 28, 28, 32) 320

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

max\_pooling2d\_1 (MaxPooling2 (None, 14, 14, 32) 0

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

conv2d\_2 (Conv2D) (None, 14, 14, 1) 289

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

max\_pooling2d\_2 (MaxPooling2 (None, 7, 7, 32) 0

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

conv2d\_3 (Conv2D) (None, 7, 7, 32) 10

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

up\_sampling2d\_1 (UpSampling2 (None, 14, 14, 32) 0

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

conv2d\_4 (Conv2D) (None, 14, 14, 32) 320

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

up\_sampling2d\_2 (UpSampling2 (None, 28, 28, 32) 0

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

conv2d\_5 (Conv2D) (None, 28, 28, 1) 289

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

activation\_1 (Activation) (None, 28, 28, 1) 0

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

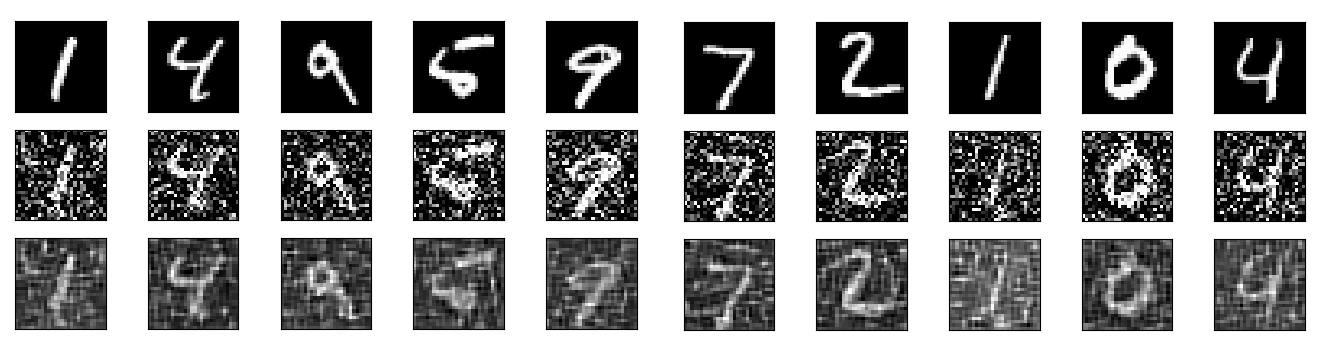


Рисунок 3. Результаты работы первой архитектуры сети

Вторая архитектура:

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

Layer (type) Output Shape Param #

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

input\_5 (InputLayer) (None, 28, 28, 1) 0

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

conv2d\_1 (Conv2D) (None, 28, 28, 32) 320

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

max\_pooling2d\_1 (MaxPooling2 (None, 14, 14, 32) 0

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

conv2d\_2 (Conv2D) (None, 14, 14, 1) 289

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

max\_pooling2d\_2 (MaxPooling2 (None, 7, 7, 1) 0

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

conv2d\_3 (Conv2D) (None, 7, 7, 1) 10

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

up\_sampling2d\_1 (UpSampling2 (None, 14, 14, 1) 0

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

conv2d\_4 (Conv2D) (None, 14, 14, 32) 320

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

up\_sampling2d\_2 (UpSampling2 (None, 28, 28, 32) 0

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

conv2d\_5 (Conv2D) (None, 28, 28, 1) 289

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

activation\_1 (Activation) (None, 28, 28, 1) 0

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

1. **Текст программы:**

**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
**from** keras.models **import** Model, model\_from\_json  
**from** keras.layers **import** Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, UpSampling2D, Input, LSTM, RepeatVector  
**from** keras.datasets **import** mnist  
**from** keras.utils.vis\_utils **import** plot\_model  
  
(X\_train, \_), (X\_test, \_) = mnist.load\_data()  
shape\_x = 28  
shape\_y = 28  
  
X\_ist = X\_test.astype(**'float32'**) / 255.  
X\_ist = X\_test.reshape(-1,shape\_x,shape\_y,1)  
  
X\_train=X\_train.astype(**'float32'**)/255  
X\_test=X\_test.astype(**'float32'**)/255  
X\_train=np.reshape(X\_train,(len(X\_train),28,28,1))  
X\_test=np.reshape(X\_test,(len(X\_test),28,28,1))  
noise\_factor = 0.5  
X\_train = X\_train + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=X\_train.shape)  
X\_test = X\_test + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=X\_test.shape)  
X\_train = np.clip(X\_train, 0., 1.)  
X\_test = np.clip(X\_test, 0., 1.)  
  
n=5  
plt.imshow(X\_ist[3].reshape(28,28))  
plt.gray()  
plt.show()  
**for** i **in** range(n):  
 plt.imshow(X\_test[i].reshape(28,28))  
 plt.gray()  
 plt.show()  
 input\_img = Input(shape=(shape\_x, shape\_y, 1))  
  
*# Ecoding*x = Conv2D(32, (3, 3), padding=**'same'**, activation=**'relu'**)(input\_img)  
x = MaxPooling2D(pool\_size=(2,2), padding=**'same'**)(x)  
x = Conv2D(16,(3, 3), padding=**'same'**, activation=**'relu'**)(x)  
encoded = MaxPooling2D(pool\_size=(2,2), padding=**'same'**)(x)  
  
*# Decoding*x = Conv2D(16,(3, 3), padding=**'same'**, activation=**'relu'**)(encoded)  
x = UpSampling2D((2, 2))(x)  
x = Conv2D(32,(3, 3), padding=**'same'**, activation=**'relu'**)(x)  
x = UpSampling2D((2, 2))(x)  
x = Conv2D(1,(3, 3), padding=**'same'**)(x)  
  
decoded = Activation(**'linear'**)(x)  
  
autoencoder = Model(input\_img, decoded)  
autoencoder.compile(optimizer=**'adadelta'**, loss=**'mean\_squared\_error'**)  
autoencoder.summary()  
plot\_model(autoencoder, to\_file=**'autoencoder\_plot.png'**, show\_shapes=**True**, show\_layer\_names=**True**)  
  
autoencoder.fit(X\_train, X\_train, epochs = 20, batch\_size = 64, validation\_split = 0.1)  
  
*# Save autoencoder weight*json\_string = autoencoder.to\_json()  
autoencoder.save\_weights(**'autoencoder.h5'**)  
open(**'autoencoder.h5'**, **'w'**).write(json\_string)  
  
encoder = Model(inputs = input\_img, outputs = encoded)  
plot\_model(encoder, to\_file=**'encoder\_plot.png'**, show\_shapes=**True**, show\_layer\_names=**True**)  
X\_train\_enc = encoder.predict(X\_train)  
  
encoded\_imgs = encoder.predict(X\_test)  
decoded\_imgs = autoencoder.predict(X\_test)  
  
n = 10  
plt.figure(figsize=(20, 4))  
  
**for** i **in** range(n):  
 *# display original* ax = plt.subplot(3, n, i + 1)  
 plt.imshow(X\_ist[i].reshape(28, 28))  
 plt.gray()  
 ax.get\_xaxis().set\_visible(**False**)  
 ax.get\_yaxis().set\_visible(**False**)  
  
 *# Encoded images* ax = plt.subplot(3, n, i + 1 + n)  
 plt.imshow(X\_test[i].reshape(28, 28))  
 plt.gray()  
 ax.get\_xaxis().set\_visible(**False**)  
 ax.get\_yaxis().set\_visible(**False**)  
  
  
 *# display reconstruction* ax = plt.subplot(3, n, i + 1 + 2\*n)  
 plt.imshow(decoded\_imgs[i].reshape(28, 28))  
 plt.gray()  
 ax.get\_xaxis().set\_visible(**False**)  
 ax.get\_yaxis().set\_visible(**False**)  
  
plt.show()