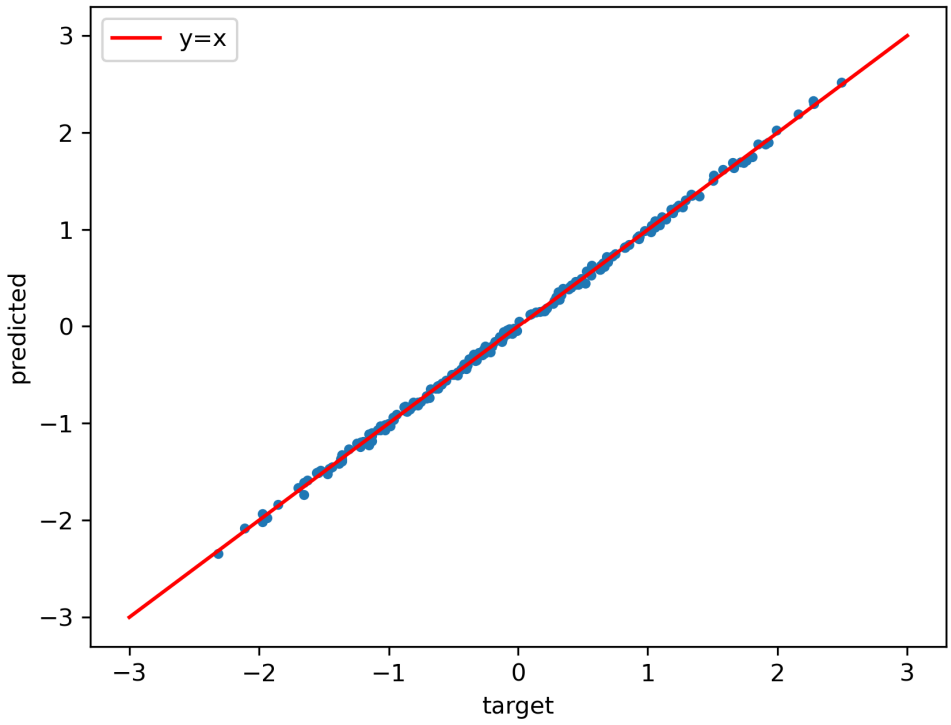
بخش اول)

در این بخش شبکه ما باید یک تابع تک بعدی را اموزش ببیند.
برای این کار ابتدا ماتریس x ای را n در t را میسازیم که n تعداد داده های ما و t تعداد ویژگی های هر داده است.
ما به صورت رندوم این ماتریس را پر می کنیم و برای بدست اوردن ماتریس خروجی،ماتریس x را به تابع نوشته شده می دهیم.

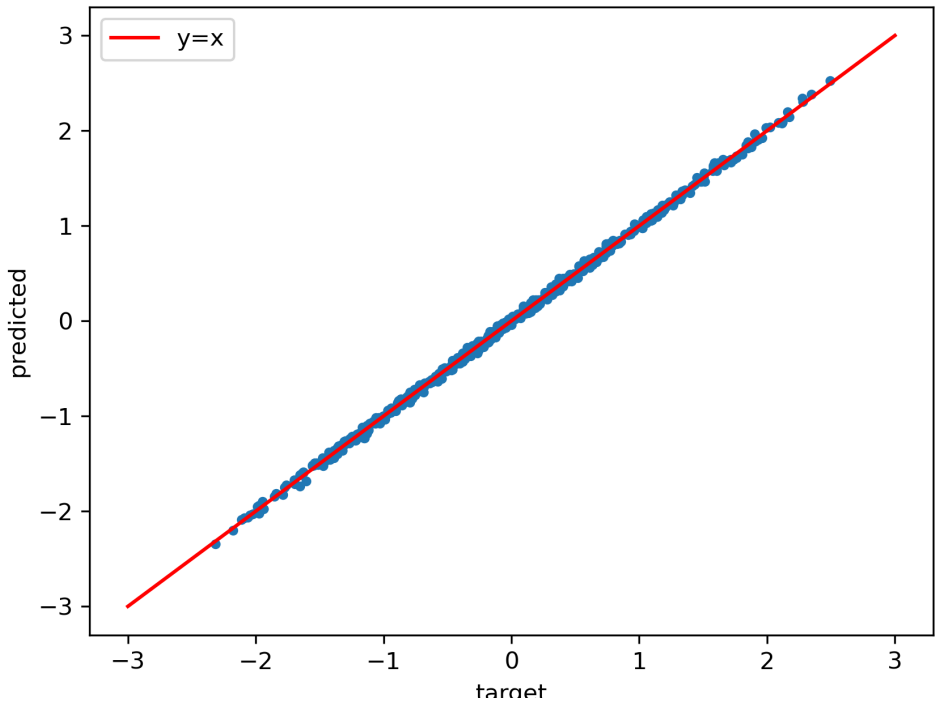
```
nD = 200
nX = 3
X = np.zeros((nD,nX))
Y = np.zeros((nD,1))


for i in range(nD):
    X [i, :] = np.random.uniform(-1,1,nX)
    Y [i , 0] = 1.2*X[i ,0] + 0.75*np.sin(X[i,1]) + np.sin(X[i,2])
```

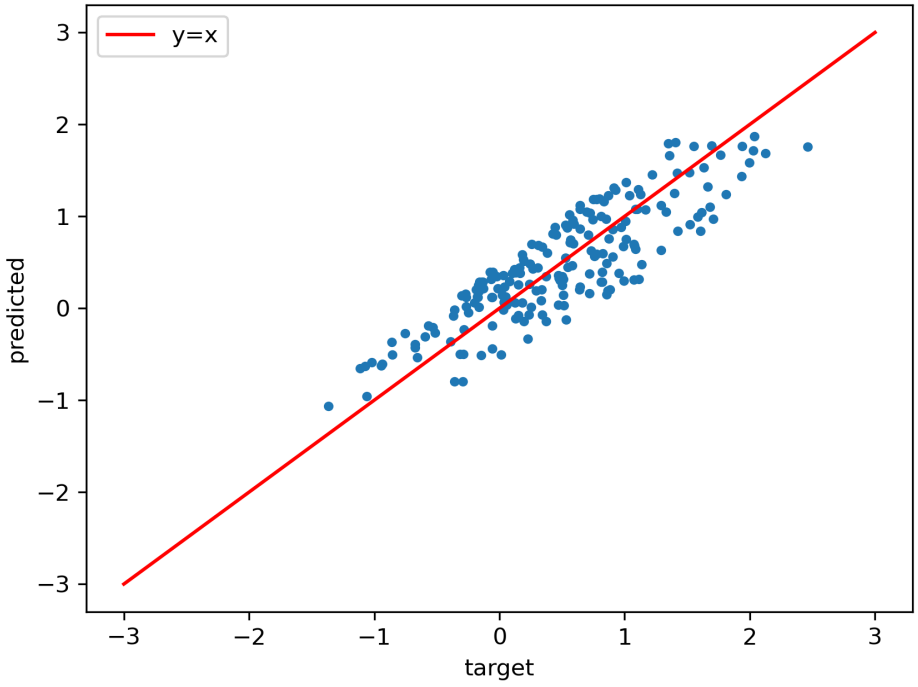
با 200 داده خطا برابر 0.00089 است و نمودار حاصل:



با 500 داده خطا برابر 0.00094 شد و نمودار:



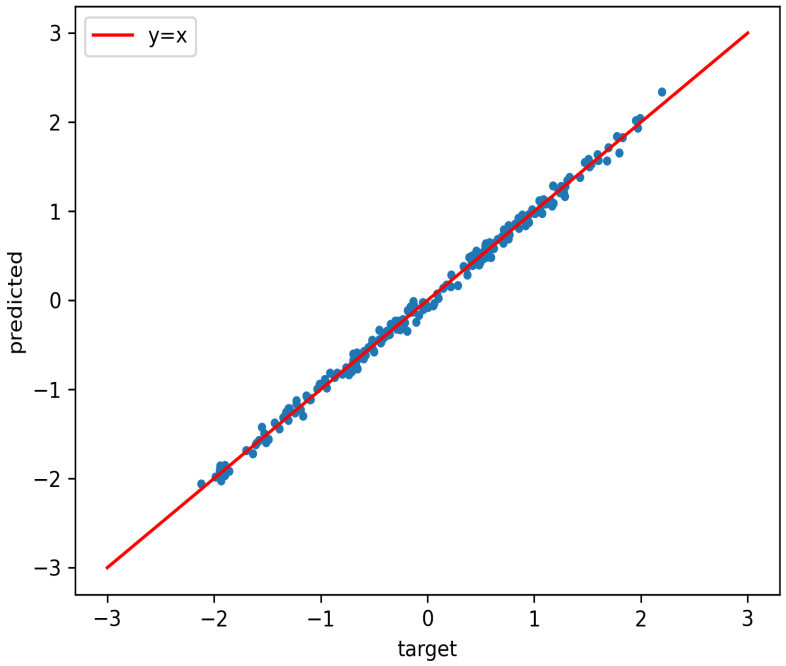
با پیچیده تر شدن تابع خطا بسیار افزایش پیدا کرد و به 0.12 رسید.



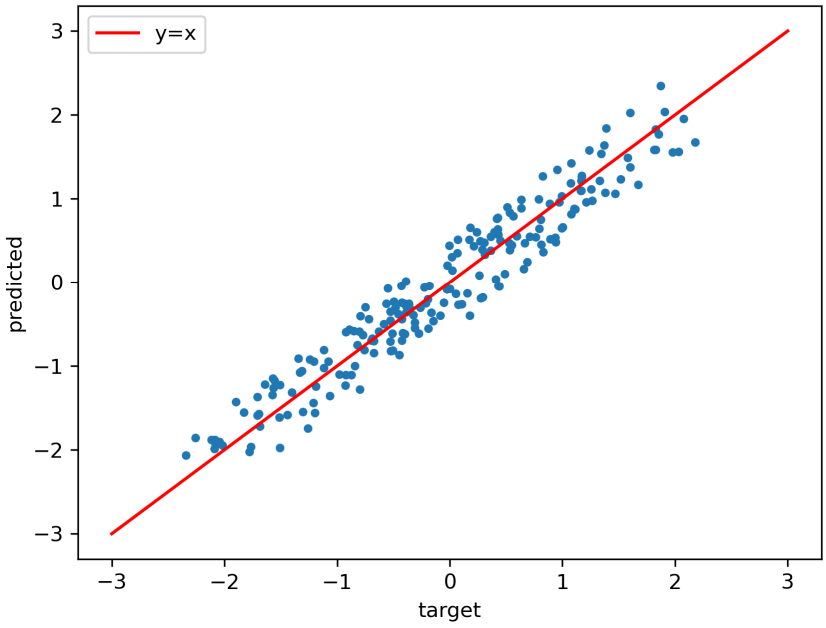
بخش دوم)

این بخش کاملا مشابه بخش اول است فقط باید به خروجی مقداری نویز اضافه کنیم.

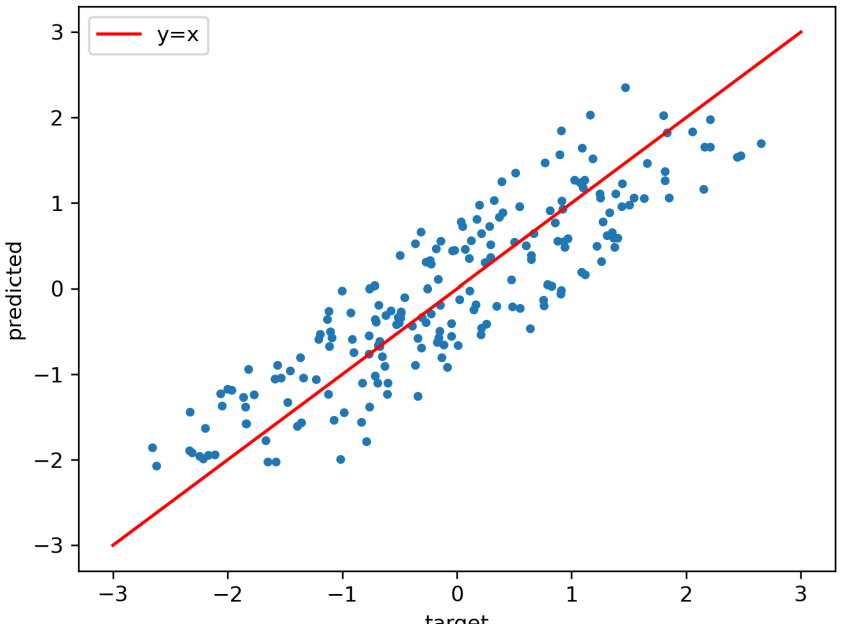
با نویز بین -0.1 تا 0.1 خطا برابر 0.0038 است.



با نویز بین -0.5 تای 0.5 خطا برابر 0.077 شد.



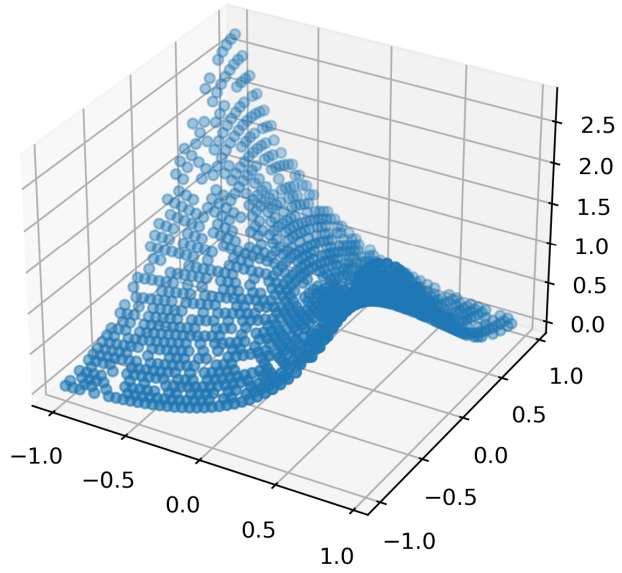
با نویز بین -1 تا 1 خطا برابر 0.308 شد.



بخش سوم)

این بخش هم مشابه بخش اول است با این تفاوت که تابع دوبعدی است.

```
def z(x,y):
    return np.exp(-(2*x*y + np.square(y)))
```



با تغییر دادن عداد لایه ها دقت را مورد بررسی قرار میدهیم.
با 20 لایه خطا برابر 0.0009 است.

```
mlp = MLPRegressor(
    hidden_layer_sizes=[20],
    max_iter=3000,
    tol=0,
)
```

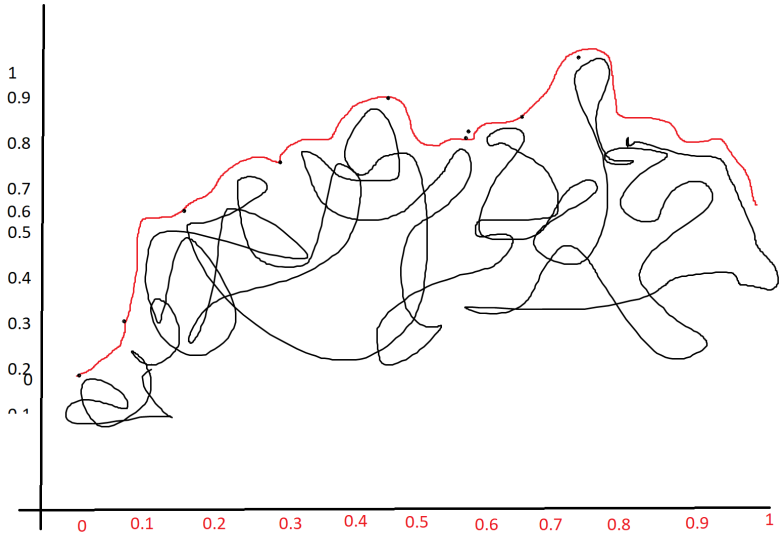
با 10 لایه خطا برابر :

```
0.004518849468548449
```

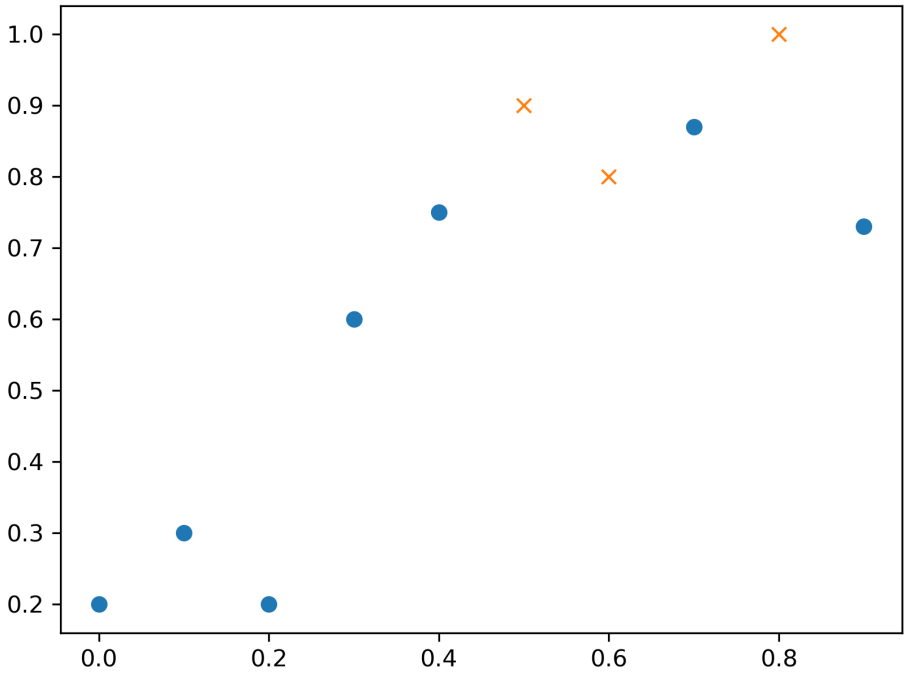
با کاهش max_iter به 100 میزان خطا بسیار افزایش پیدا کرد.

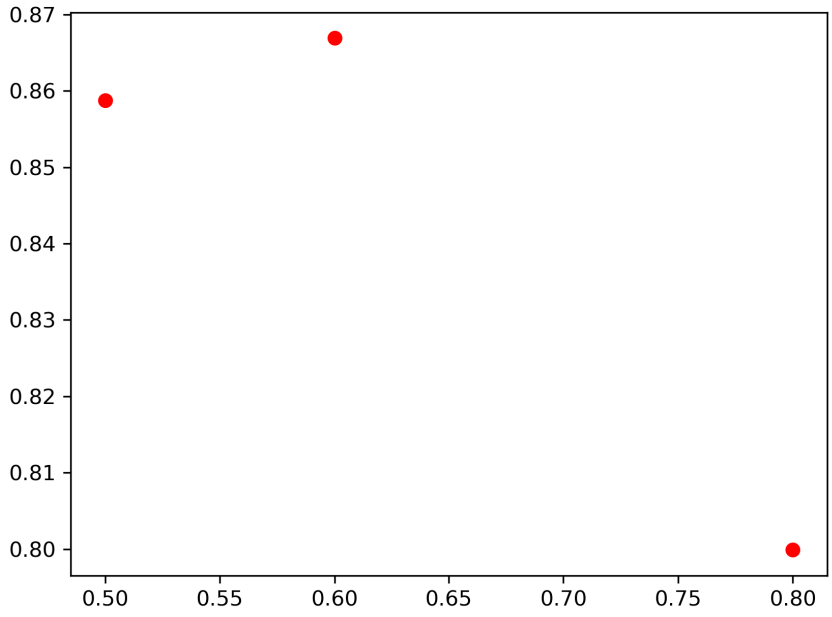
```
0.025288167507512442
```

بخش چهارم)



نقاطی از تابع بالا را به شبکه می دهیم.
ابی ها نقطه های داده شده است قرمز ها را داده تست در نظر گرفته شده.





میبینیم در جاهایی که تابع پرش کرده خطا افزایش میابد.

بخش پنجم)

در این بخش از ما خواسته شده که data set ای را انتخاب کنیم و به شبکه عصبی آن را آموزش دهیم.
ابتدا باید فایل زیپ را باز و بخوانیم.

```
[9]
    zip_address = '/content/USPS_images.zip'
    zip_ref = zipfile.ZipFile(zip_address, 'r')
    zip_ref.extractall('/content/')
    train_dir = '/content/USPS_images/train'
    validation_dir = '/content/USPS_images/test'
```

عکس های مربوط به train را در train_dir و عکس های مربوط به test را در validation_dir میریزیم.

در مرحله بعد باید شبکه را آموزش دهیم.یعنی عکس های بخش train را به عنوان x_train و جواب آن ها که با استفاده از اسم فایل عکس استخراج کردیم به y_train می دهیم.
سپس همین کار را برای داده های test نیز انجام میدهیم.
با استفاده از دستور imread عکس را تبدیل به آرایه میکنیم و سپس آن را از RGB به GRAY تبدیل میکنیم تا آرایه دوبعدی شود.

```
for path in os.listdir(train_dir):
    if os.path.isfile(os.path.join(train_dir, path)):
        y_train.append(int(path[0]))

for path in os.listdir(validation_dir):
  if os.path.isfile(os.path.join(validation_dir, path)):
      y_test.append(int(path[0]))


for path in os.listdir(train_dir):
  if os.path.isfile(os.path.join(train_dir,path)):
    toArr = cv2.imread(f"{train_dir}/{path}")
    x_train.append(cv2.cvtColor(toArr, cv2.COLOR_RGB2GRAY))

for path in os.listdir(validation_dir):
  if os.path.isfile(os.path.join(validation_dir,path)):
    toArr = cv2.imread(f"{validation_dir}/{path}")
    x_test.append(cv2.cvtColor(toArr, cv2.COLOR_RGB2GRAY))
```

در این بخش از کد، تابع to_categorical اعداد y_train که بین 0 تا 9 هستند را بصورت آرایه ای از 0 و 1 میکند،به این صورت که ایندکس nام آرایه 1 و بقیه اعضا آن 0 است.

```
x_train = np.array(x_train)
x_test = np.array(x_test)

y_train_cat = to_categorical(np.array(y_train), 10)
y_test_cat = to_categorical(np.array(y_test), 10)
```

```
x_train_final = x_train.reshape(-1 ,16*16) / 255
x_test_final = x_test.reshape(-1 ,16*16) / 255
```

```
model = Sequential()
model.add(Input(shape = (16*16)))
model.add(Dense(20, activation = 'relu'))
model.add(Dense(20, activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(10 , activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam' , metrics = ['accuracy'])
```

در این بخش،اعداد x_train که بصورت ماتریسی از اعداد بین 0 تا 256 است را،به اعدا بیت 0 و 1 تبدیل میکند.
در این قسمت مدل مورد نظر را با استفاده از کتابخانه keras می سازیم.سپس به تعداد دلخواه لایه اضافه می کنیم.تعداد نورون لایه ها را هم به دلخواه انتخاب میکنیم.لایه اخر،لایه خروجی ما است و چون خروجی ما باید عددی بین 0 تا 9 باشد،تعداد نورون آن لایه را 10 می گذاریم.تابع فعال سازی لایه آخر را softmax میگذاریم زیرا این تابع به گونه ای عمل میکند که مجموع احتمال تمامی نورون های لایه آخر برابر 1 شود و خروجی عددی است که بیشترین احتمال را دارد.

```
[14] batch_size = 128
     epochs = 30
     model.fit(x_train_final, y_train_cat,
               batch_size= batch_size ,
               epochs=epochs, verbose= 1,
               validation_data=(x_test_final,y_test_cat))
```

در کد بالا،batch_size تعیین میکند که هر بار چه تعداد داده به لایه بعد فرستاده شود و epochs تعیین میکند این کار چند بار اجرا شود.

بخش ششم)

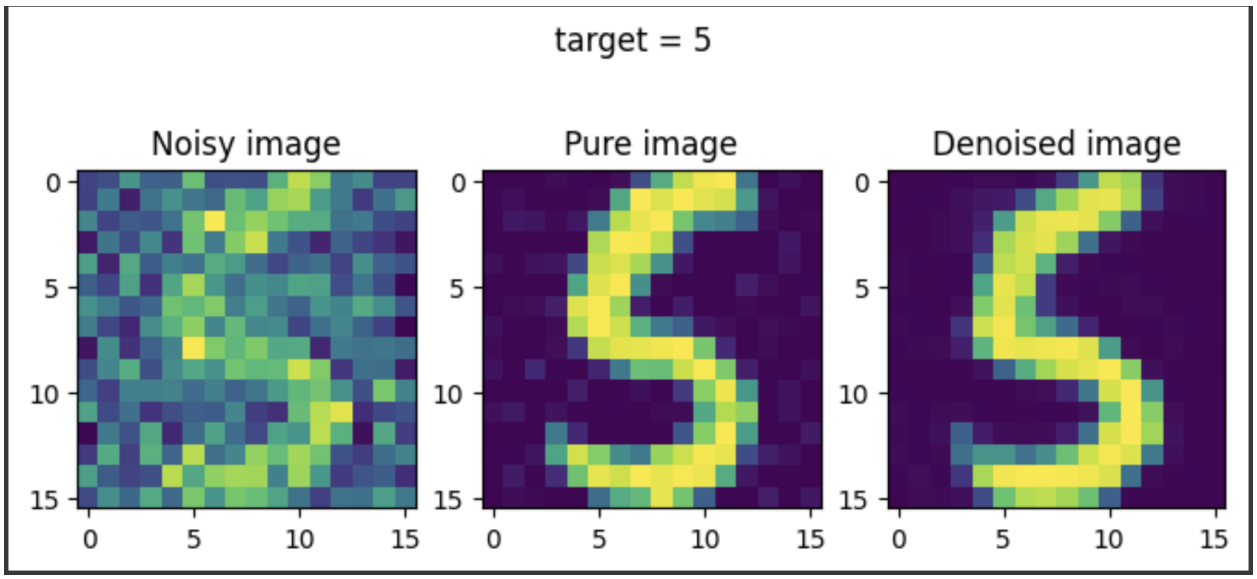
در این بخش کار های اولیه بخش پنجم را انجام می دهیم.یعنی باز کردن و خواندن داده ها و ریختن آن ها در لیست های جداگانه.
حال باید ورودی شبکه را عکس نویز دار و خروجی آن را عکس بدون نویز بدهیم.
برای این کار ابتدا به داده های ورودی یعنی x_train نویز را اضافه میکنیم.

```
# Add noise
pure = x_train
pure_test = x_test
noise = np.random.normal(0, 1, pure.shape)
noise_test = np.random.normal(0, 1, pure_test.shape)
noisy_input = pure + noise_factor * noise
noisy_input_test = pure_test + noise_factor * noise_test
```

```
model.compile(optimizer='adam', loss='binary_crossentropy')
model.fit(noisy_input, pure,
                epochs=no_epochs,
                batch_size=batch_size,
                validation_split=validation_split,
                verbose=0)
```

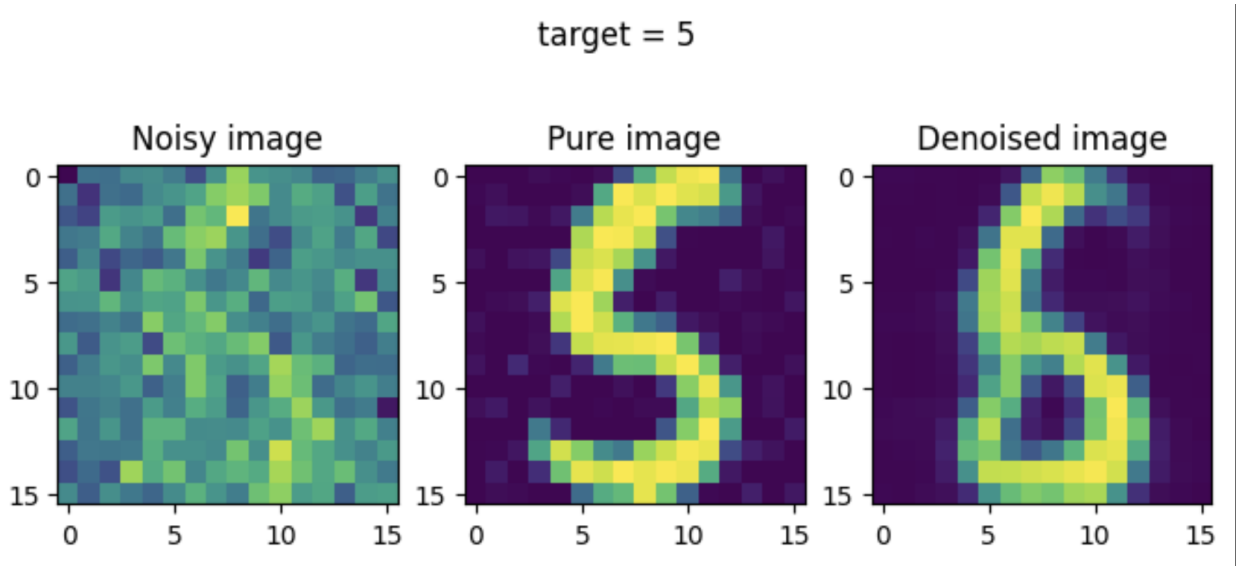
در اینجا مدلمان را میسازیم و عکس نویز دار را به عنوان ورودی و بدون نویر آن را به عنوان خروجی می دهیم تا شبکه آموزش ببیند.

با تغییر دادن مقدار noise_factor میزان نویز را از کم تا زیاد آزمایش میکنیم.



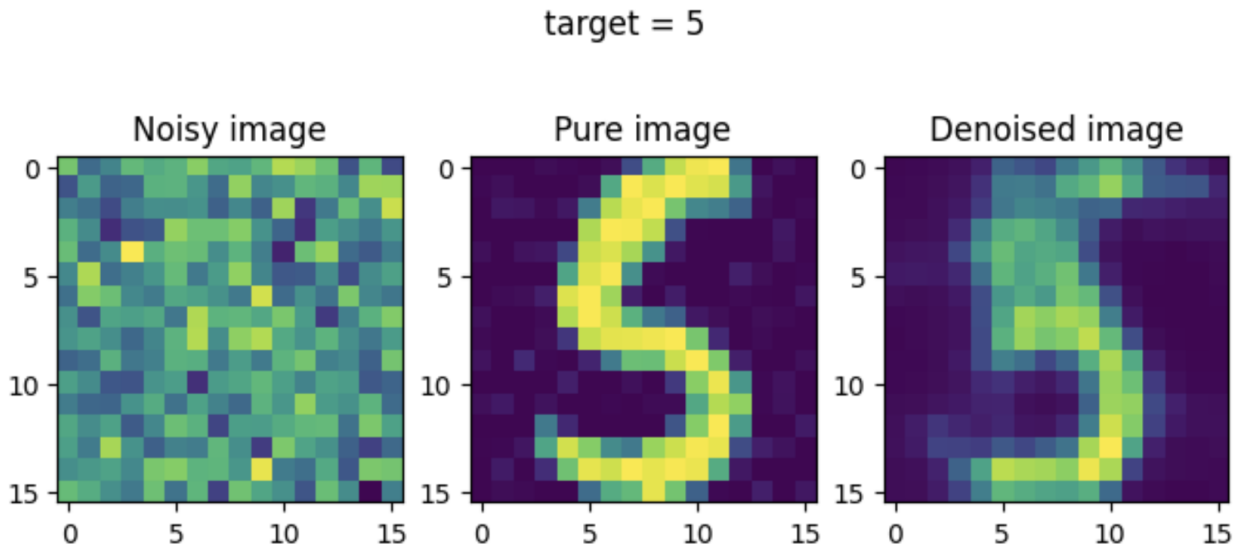
Noise factor = 0.35

شبکه با دقت نسبتا زیادی توانسته denoise کند.



Noise factor = 0.55

با این میزان نویز می بینیم که شبکه درست نتوانسته عدد مورد نظر را denoise کند.



Noise factor = 0.9

در این حالت شبکه با دقت کمی توانسته denoise کند.

از این آزمایش ها میتوانیم بفهمیم شبکه عکس ها با نویز نسبتا کم را میتواند به درستی denoise کند.