
KÜNSTLICHE INTELLIGENZ –
EIN OPTIMIERUNGSPROBLEM
HALLO MAMA!

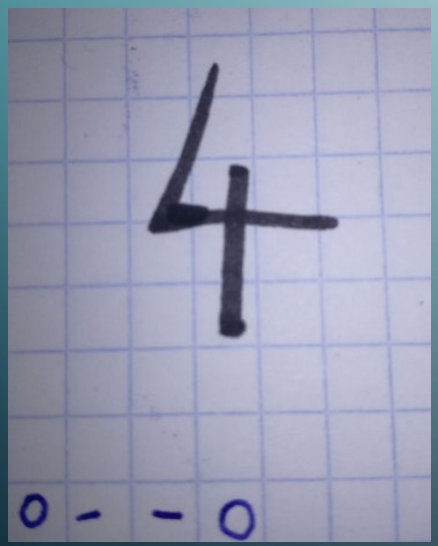
PROBLEMATIK:

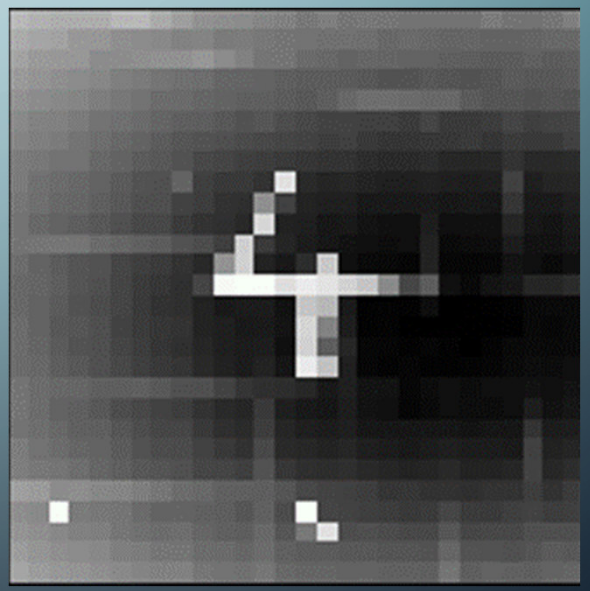
- Netzwerk soll Zahlen erkennen
- Kann Testdaten erkennen
- Gezeichnete Daten problematisch
- Lösungsansätze:

VERÄNDERUNG DER TESTDATEN

VERÄNDERN DER TESTDATEN

- Problem: Eingelesenes Foto undeutlich
  (zu viel Hintergrundrauschen u.Ä.)

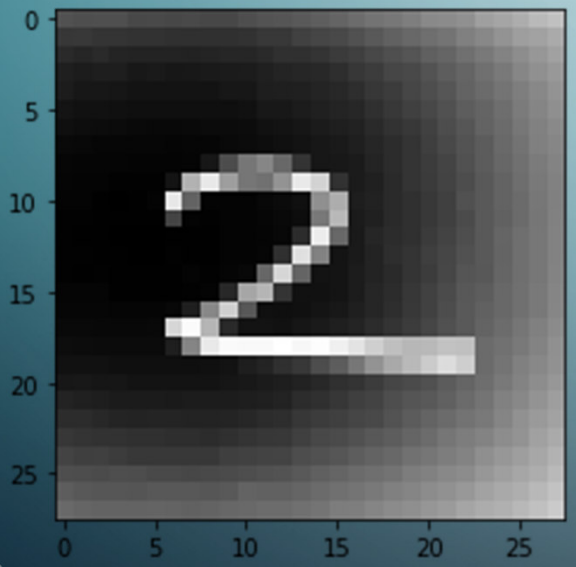




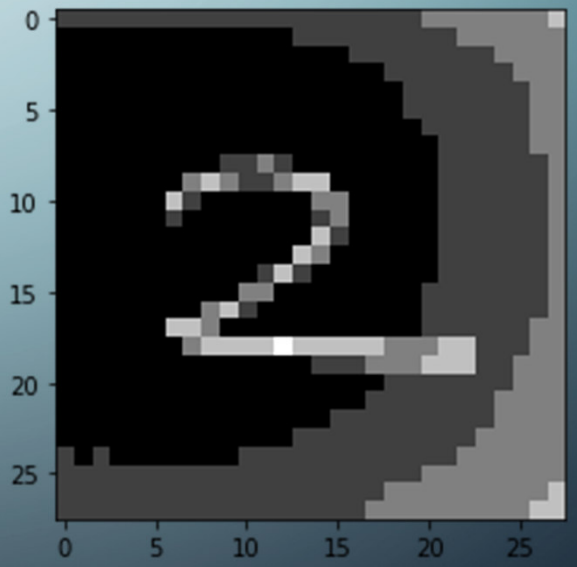
VERÄNDERN DER TESTDATEN

- Idee: weniger unterschiedliche Graustufen

Davor:



Danach:



- Weitere Idee:

  Schaffen eines klaren Kontrasts zwischen Hintergrund und Zahl

- Ausgangsbild
- Veränderung der Pixelhelligkeit nach Abstand zur Mitte
- Reskalieren der Werte von 0 bis 1

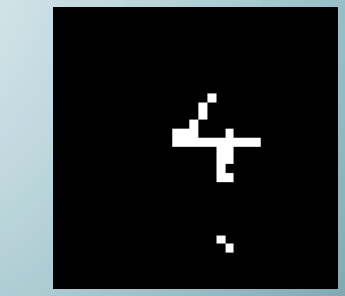
- Quadrieren der einzelnen Pixelwerte zum Erzeugen stärkerer Kontraste

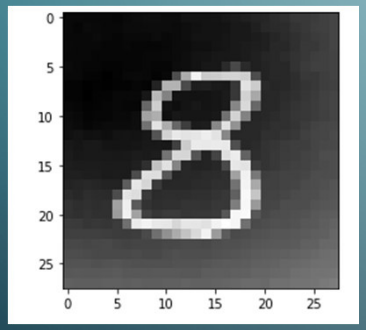
- Hervorheben besonders heller Pixel mitsamt Umgebung
- Schwärzen von dunkleren Flächen

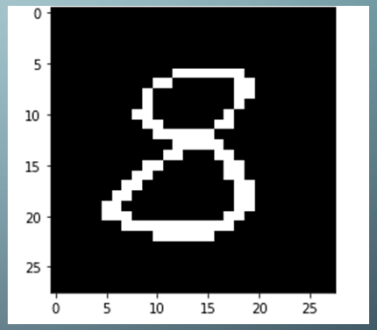
- Färben aller helleren Pixel weiß
- Schwärzen aller dunklen Pixel

ERGEBNIS

vorher: nachher:





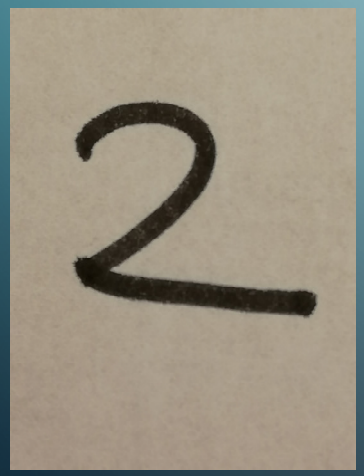


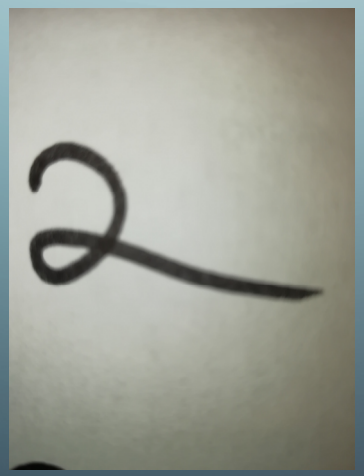
Erfolgsrate: Vor Bearbeitung: 20% der Zahlen richtig erkannt

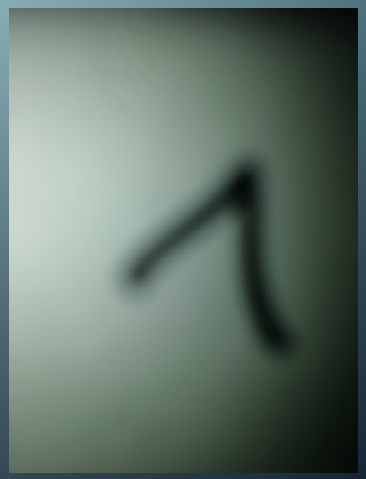
Nach Bearbeitung: 90% der Zahlen richtig erkannt

IDEE: EIGENE TRAININGSDATEN

- Handgeschriebene Einser und Zweier fotografiert
- Verschiedener Winkel/Zoom/Licht
- ~190 Fotos pro Ziffer
- ~96% Genauigkeit





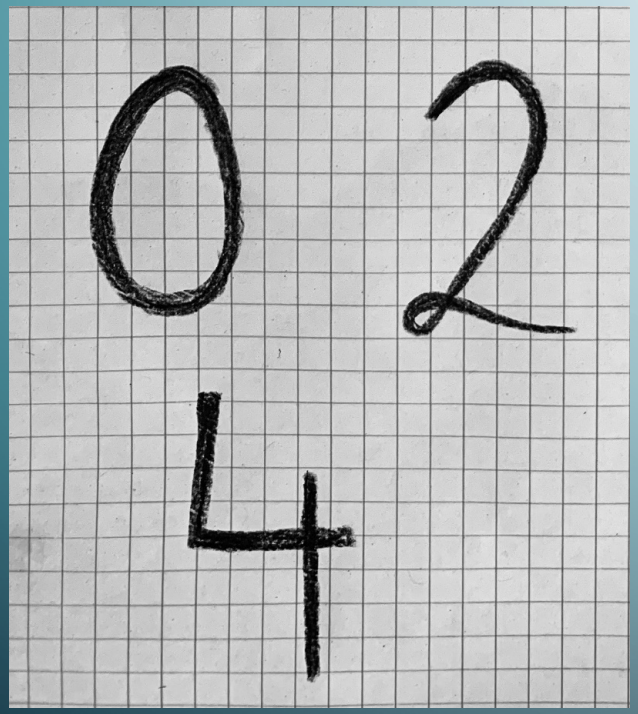


TRAININGSDATEN VERÄNDERN

TRAININGSDATEN VERÄNDERN

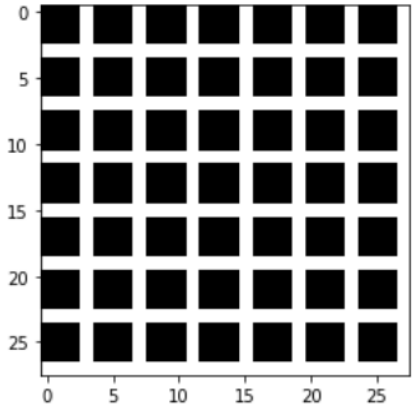
- Datenrauschen
- Raster
- Convolutional Layer
- Datenrauschen mit Kreismuster

RASTER



```
[38] karo=np.zeros((28,28))
     interval= r
     for i in range(0,28):
         for j in range(0,28):
           if (i+x)%interval==0:
             karo[i,j]=0.5
           if (j+x)%interval==0:
             karo[i,j]=0.5
           karo[i,j]=karo[i,j]
     plt.imshow(karo, cmap='gray')
```

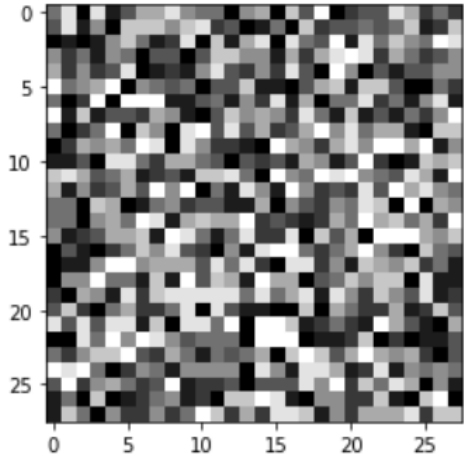
```
<matplotlib.image.AxesImage at 0x7f11caf723c8>
```



DATENRAUSCHEN

```
datenrauschen=np.zeros((28,28))
for i in range(0,28):
  for j in range(0,28):
    datenrauschen[i,j]+= random.randint(0,9)/500
plt.imshow(datenrauschen, cmap='gray')
```

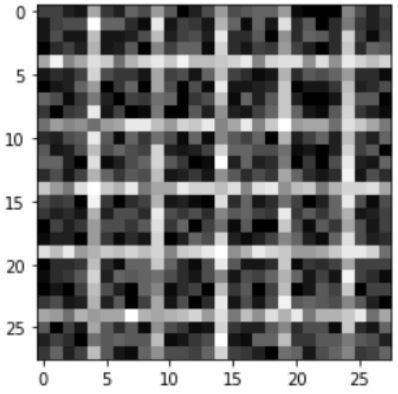
```
<matplotlib.image.AxesImage at 0x7fe9359cda20>
```



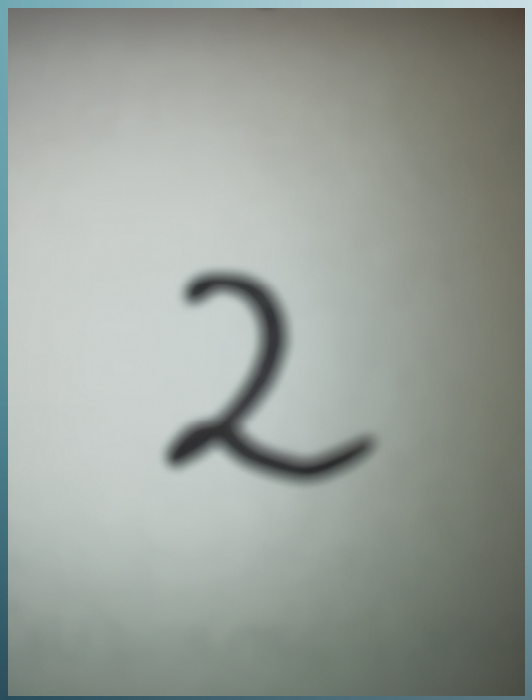

```
karo=np.zeros((28,28))
r= random.randint(4,7)
x= random.randint(0,10)
interval= r
for i in range(0,28):
    for j in range(0,28):
      if (i+x)%interval==0:
        karo[i,j]=random.randint(5,7)/50.
      if (j+x)%interval==0:
        karo[i,j]= random.randint(5,7)/50.
      karo[i,j]=karo[i,j]+random.randint(0,9)/100.

plt.imshow(karo, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f11cbc14a90>
```

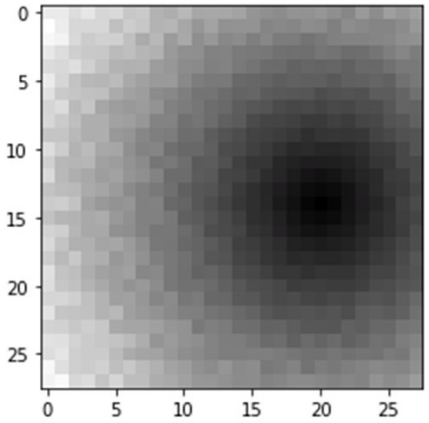


DATENRAUSCHEN MIT KREISMUSTER

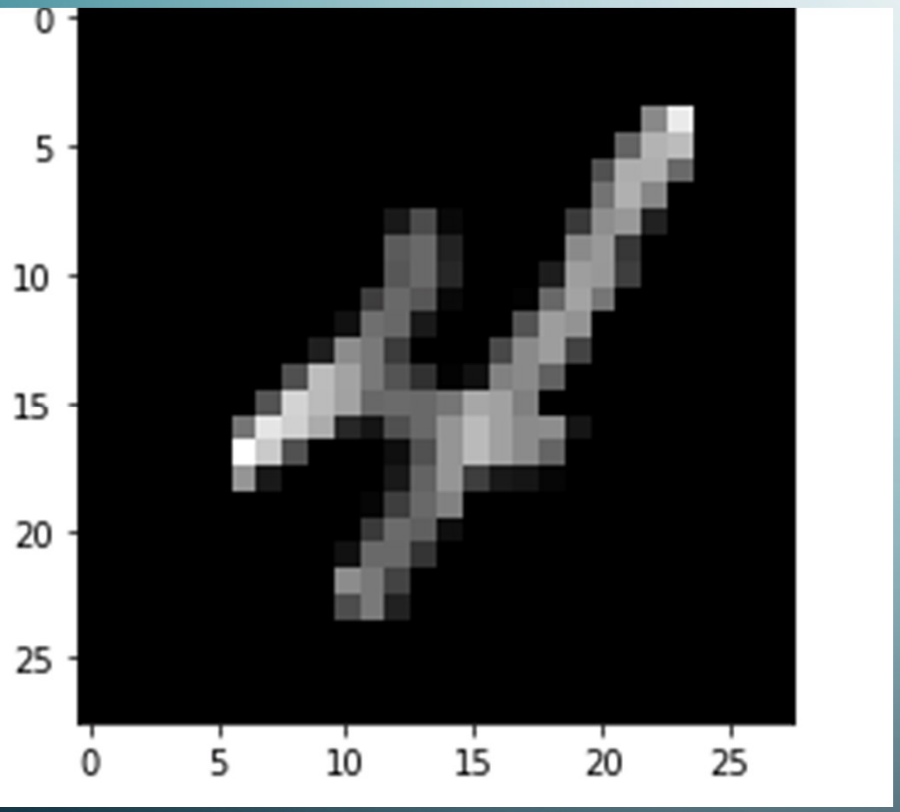


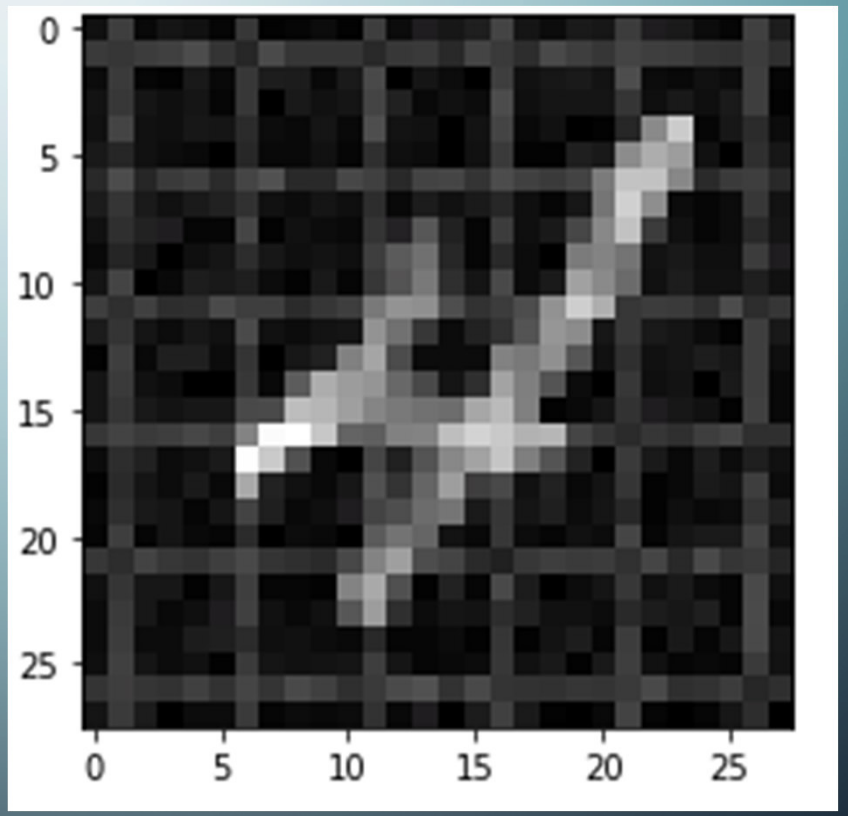
```
kreisrauschen=np.zeros((28,28))
m=20-random.randint(0,10)
n=20-random.randint(0,10)
for i in range(0,28):
  for j in range(0,28):
    r=np.sqrt(np.square((i-m))+np.square((j-n)))
    kreisrauschen[i,j]= r/random.randint(75,85)
plt.imshow(kreisrauschen,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7f23c7aa8f28>
```

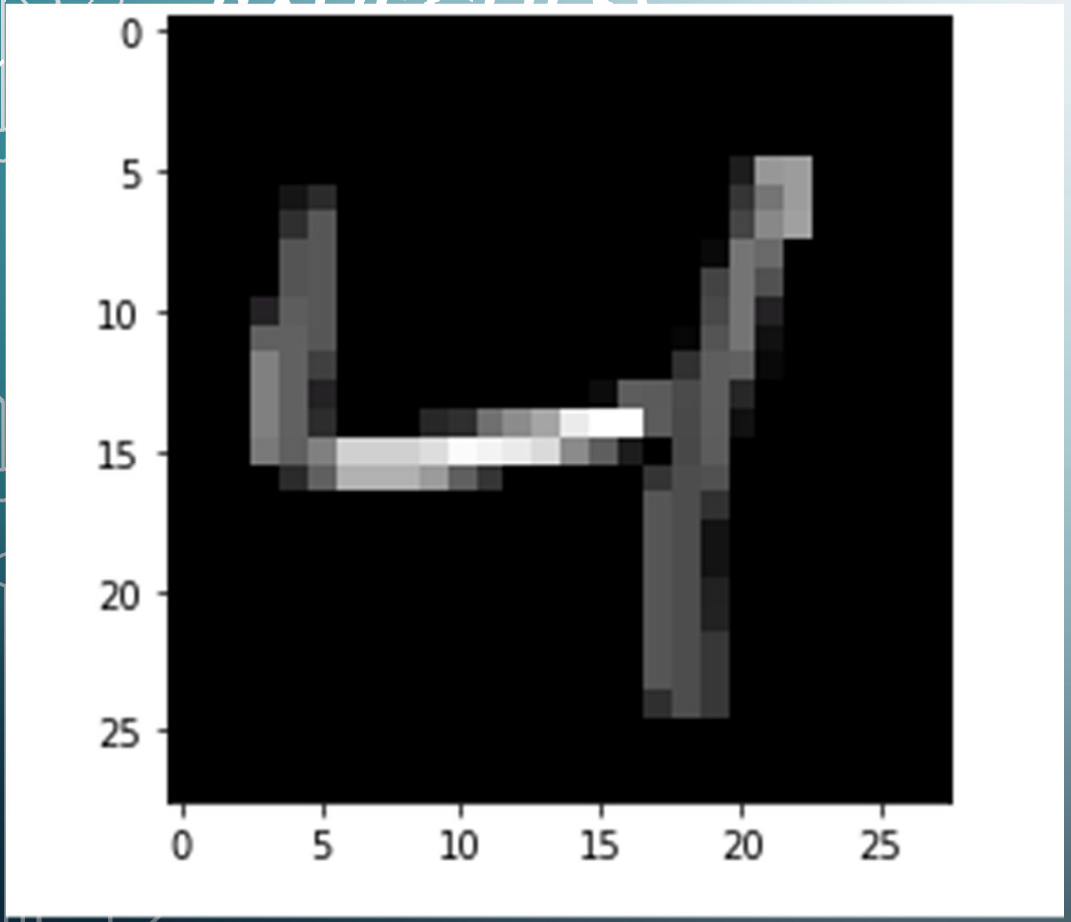


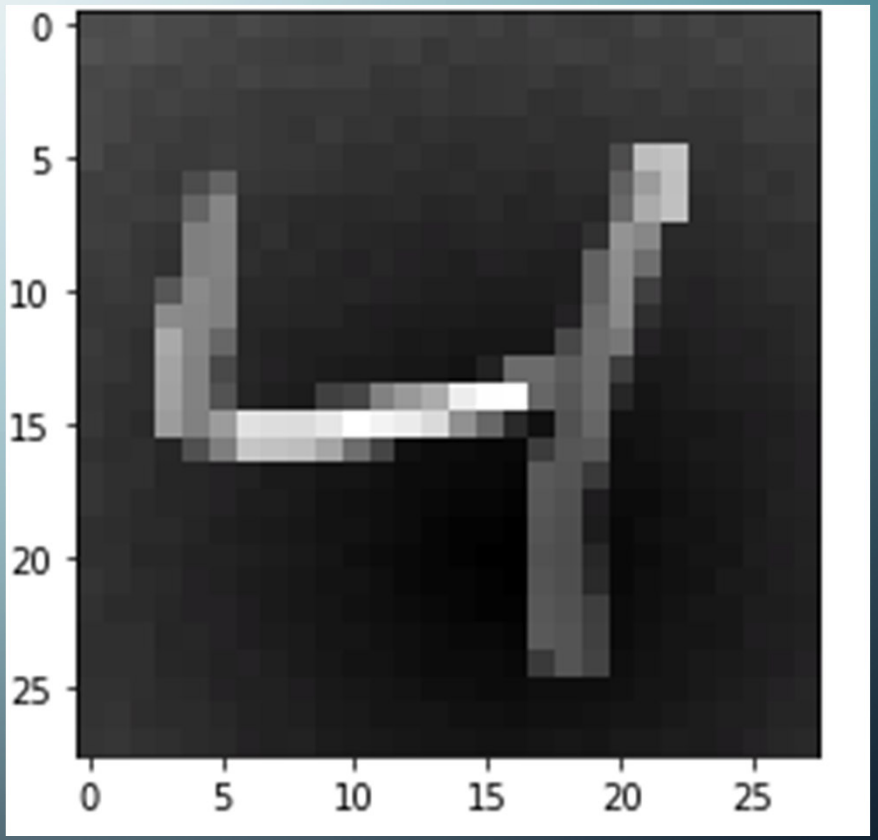
4 MIT UND OHNE RASTER



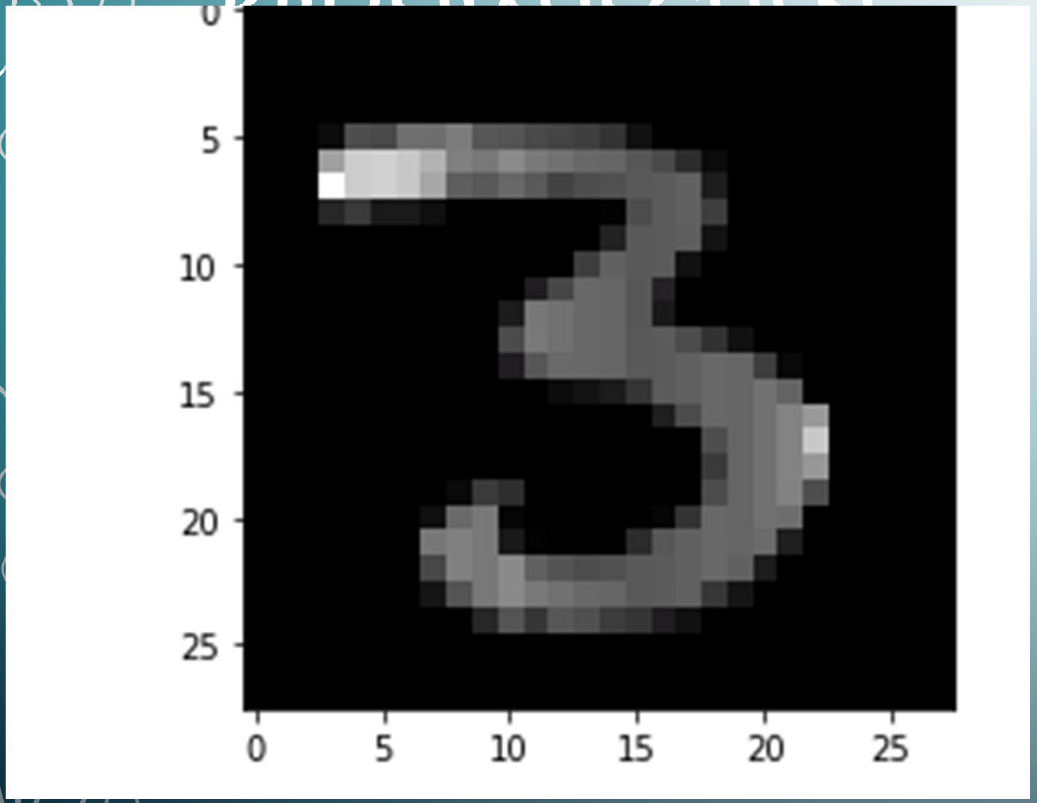


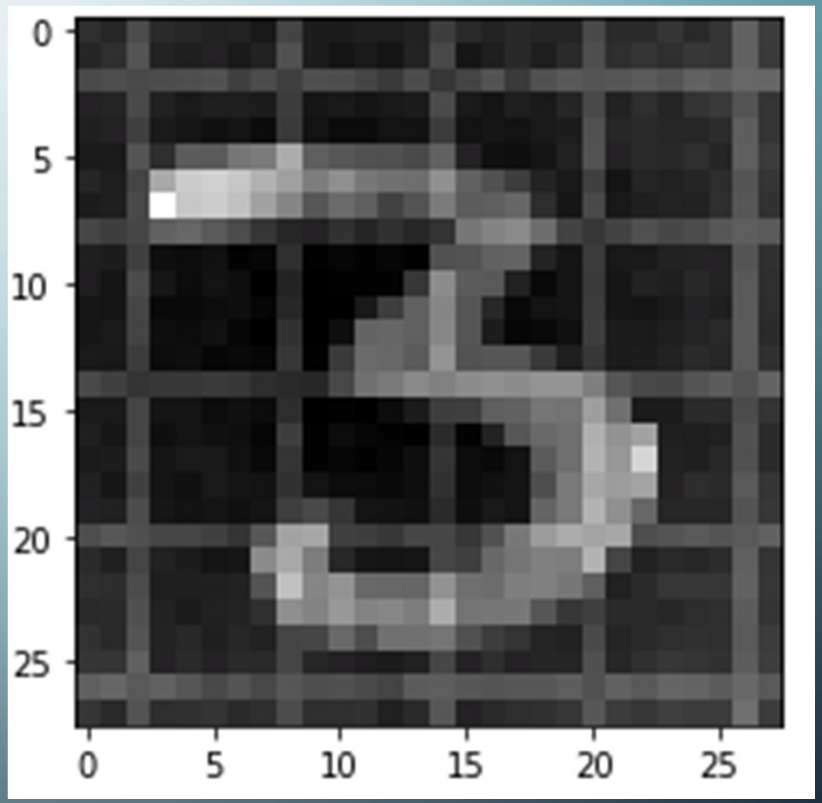
4 MIT UND OHNE KREISFÖRMIGES BAUCHEN





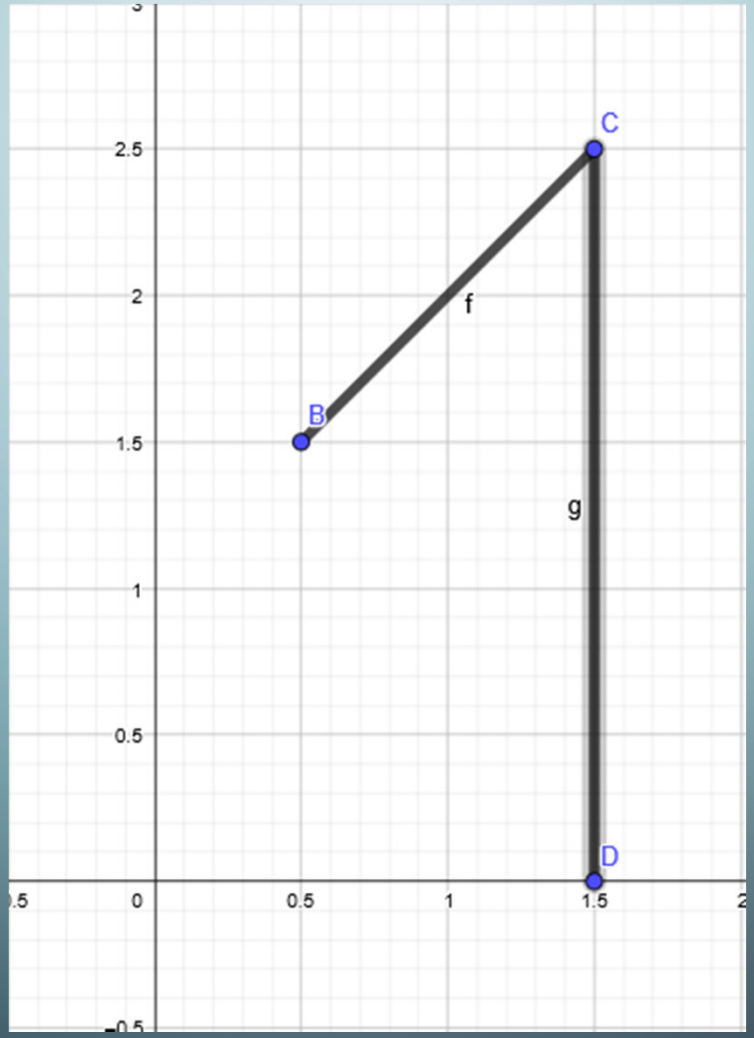
3 MIT UND OHNE RASTER + KREISRAUSCHEN

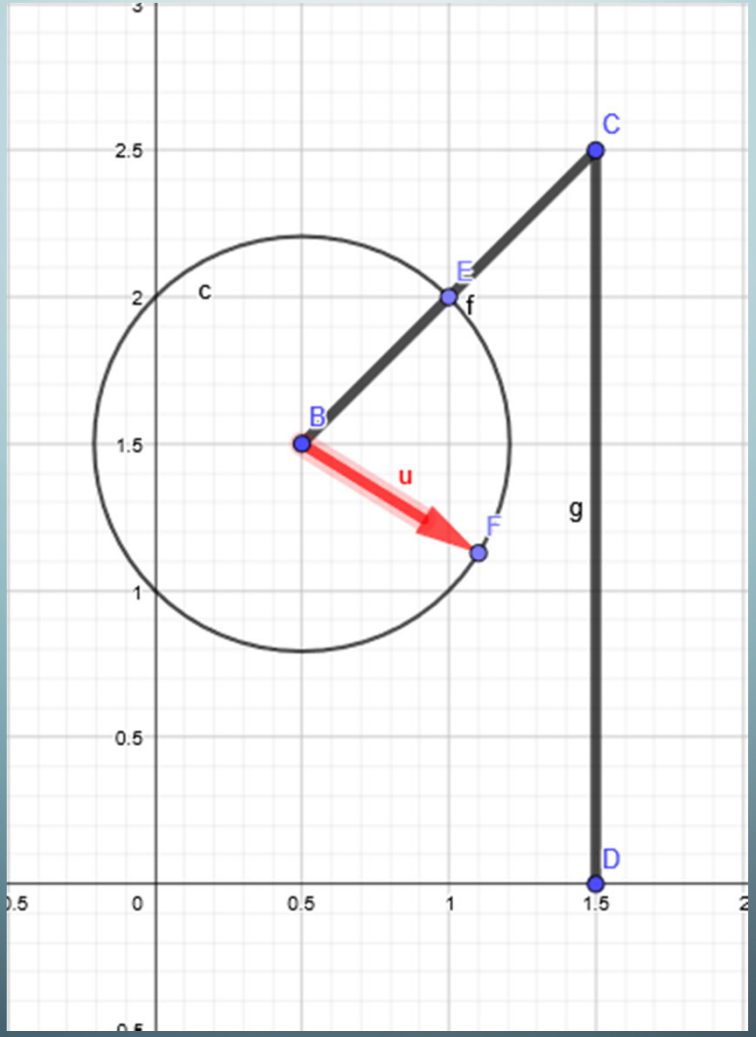


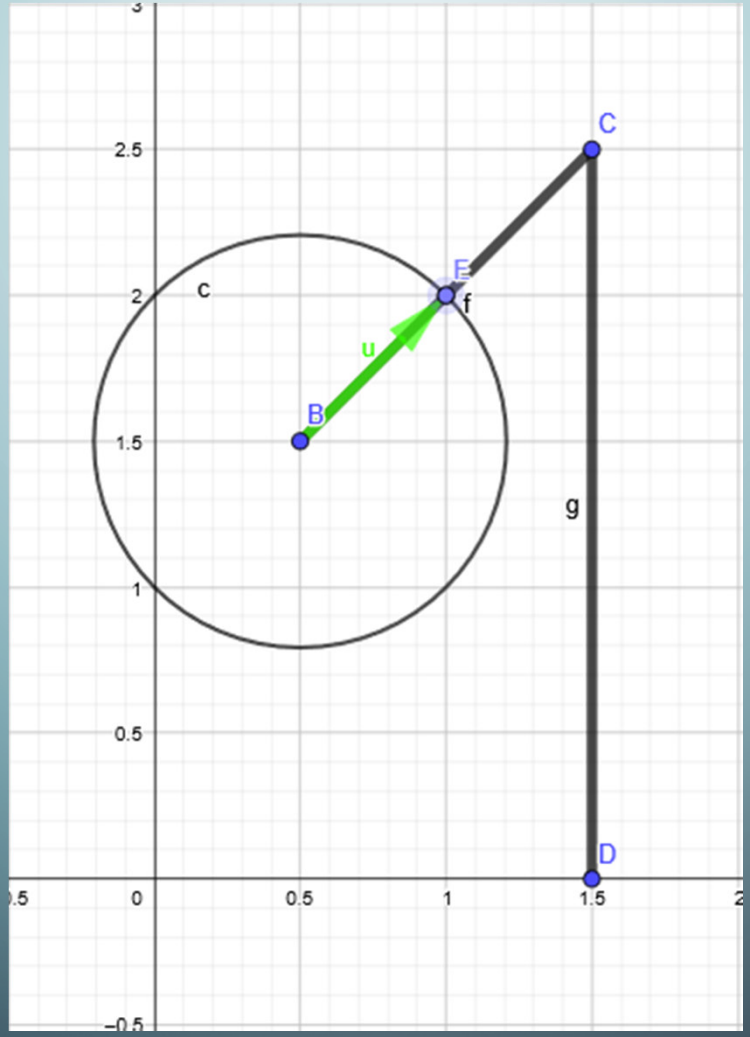


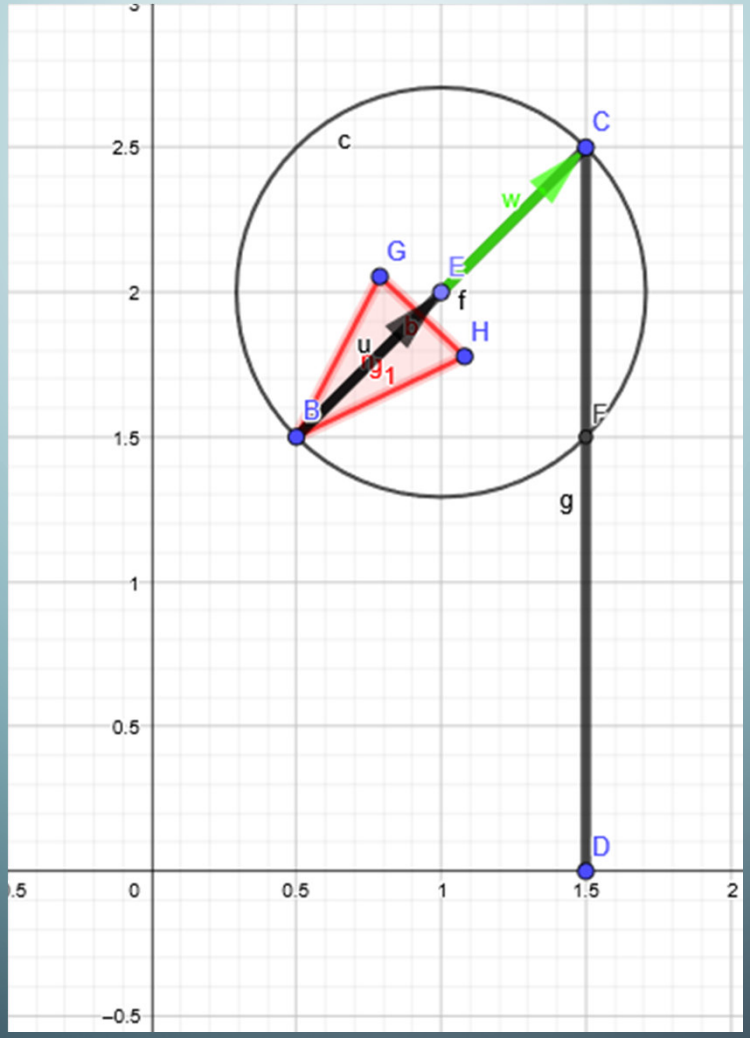
PIXEL → VEKTOREN

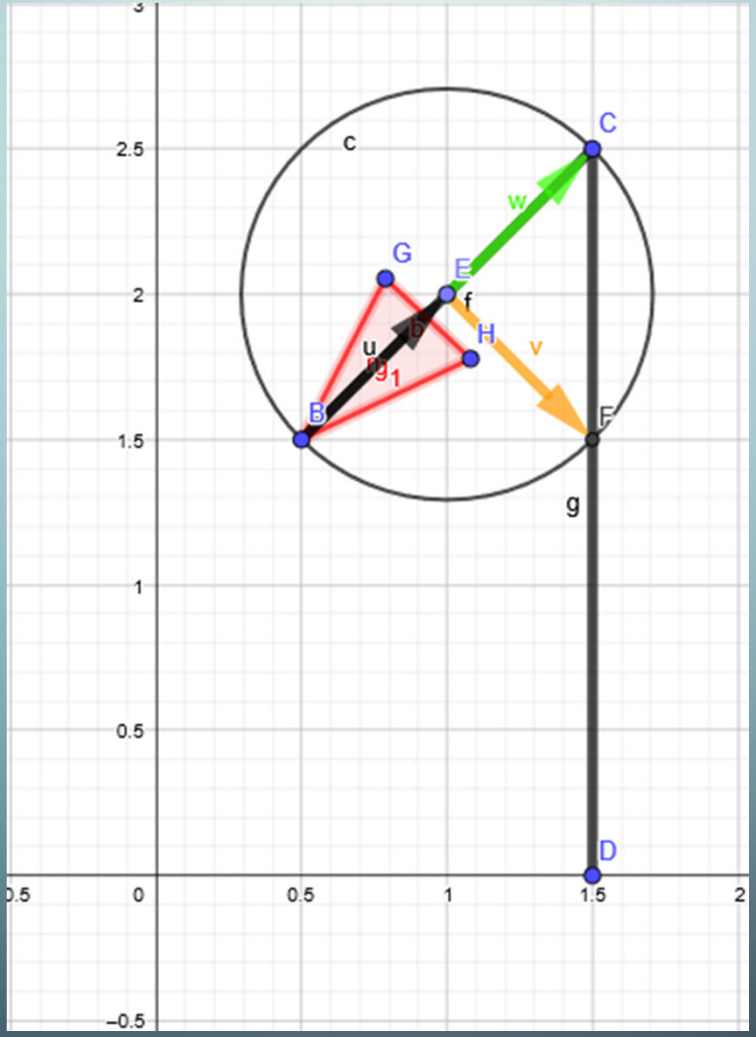

2.5
0.5
1
-0.5

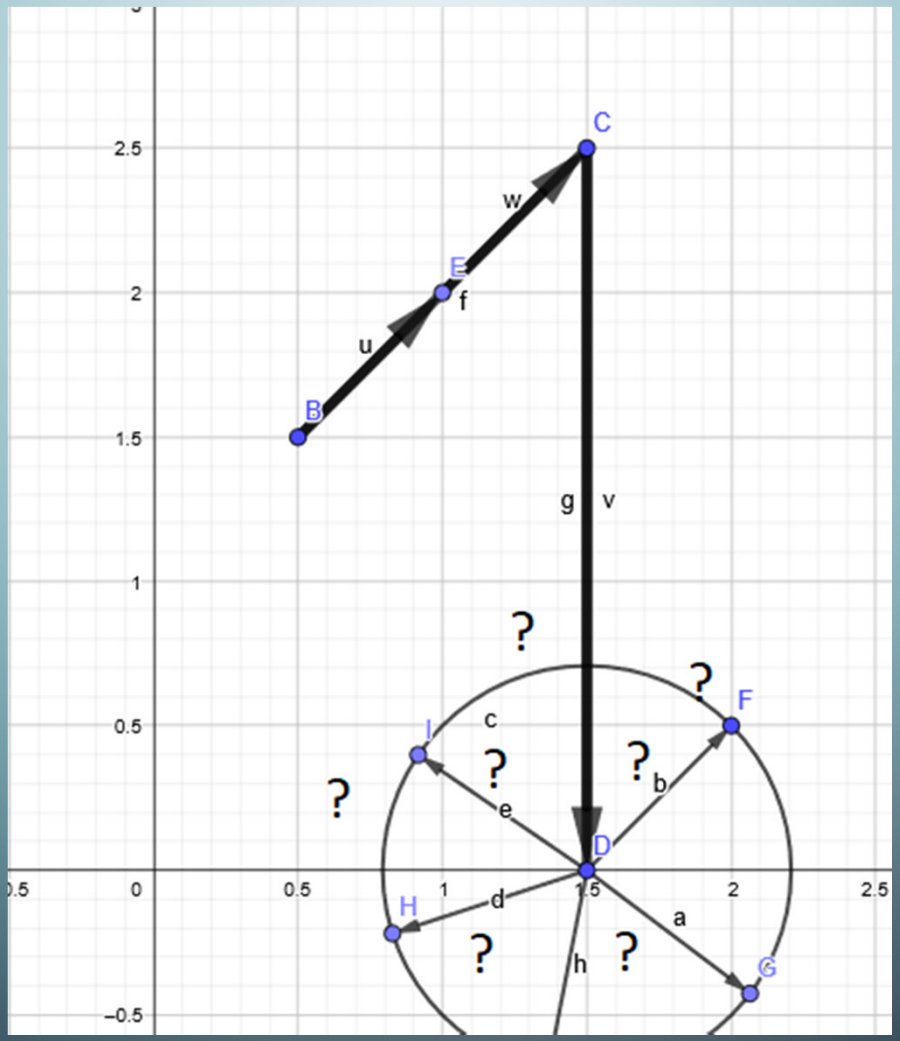

C
2.5
2
f
B
1.5
g
1
0.5
D
0
0.5
1
1.5
-0.5

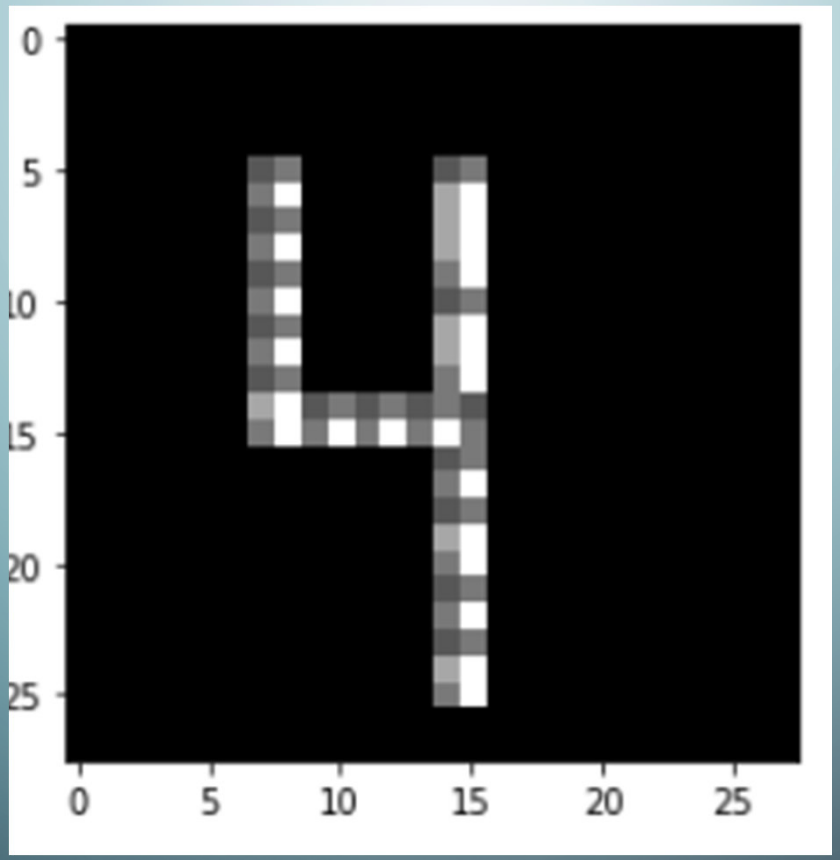

C
E
f
c
B
u
F
g
D
2.5
2
1.5
1
0.5
0
0.5
1
1.5

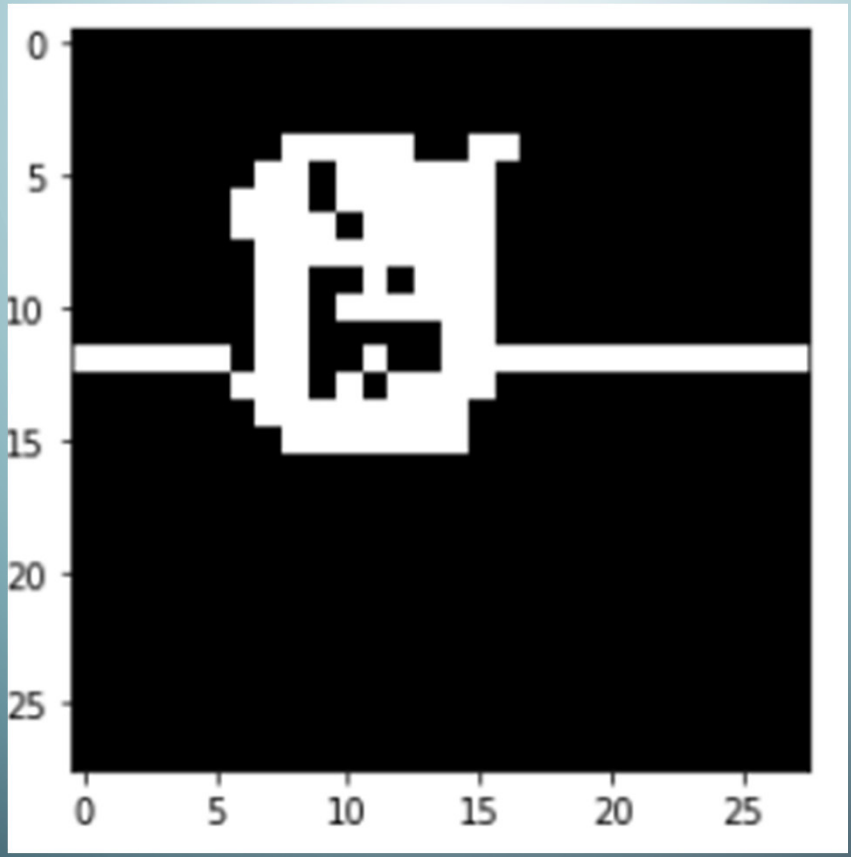

C
2.5
E
2
c
f
u
B
1.5
g
1
0.5
D
0
0.5
1
1.5
−0.5


c
C
w
G
E
f
H
u
B
F
g
D
2.5
2
1.5
1
0.5
0
−0.5
0.5
1
1.5
2

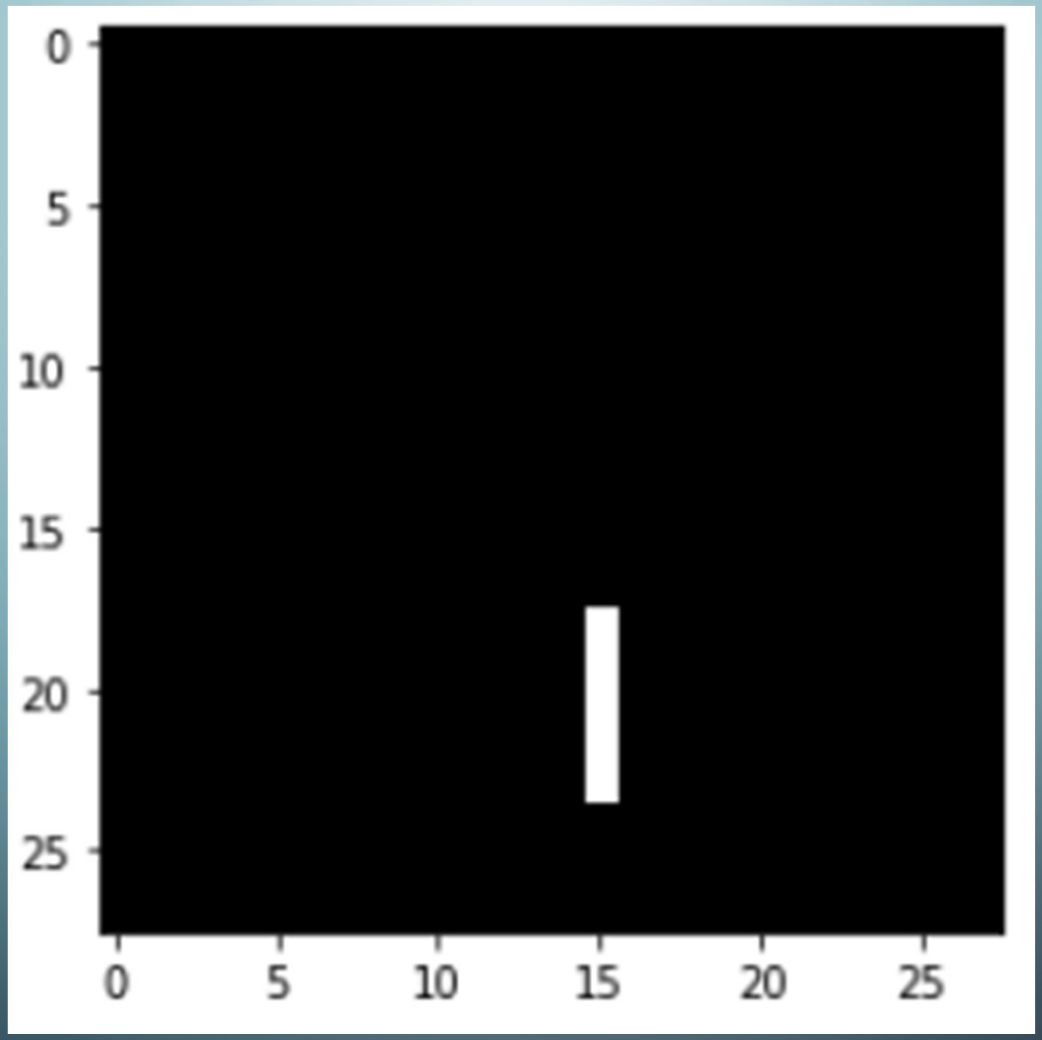

c
C
w
G
E
f
H
v
u
B
F
g
D
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
−0.5

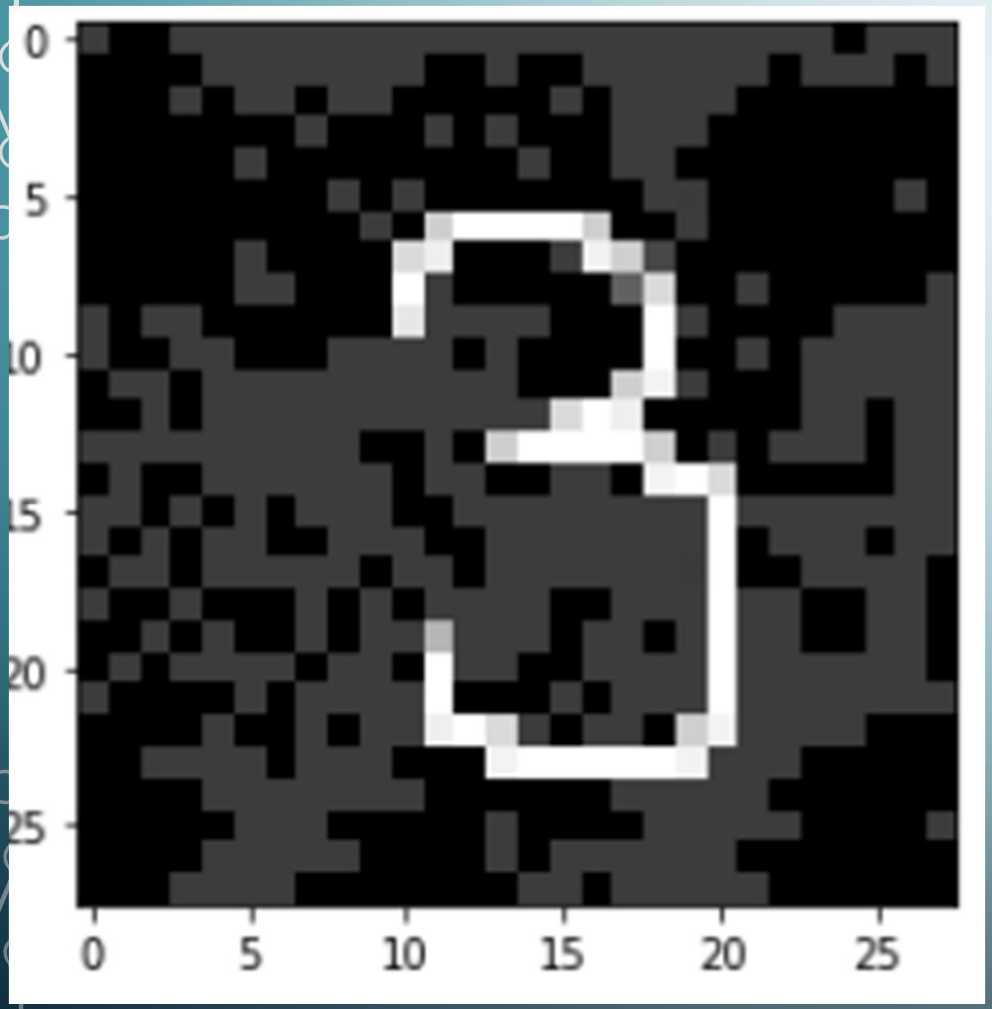

C
w
E
f
u
B
g
v
?
?
?
?
?
F
c
I
b
e
D
H
d
a
h
G
2.5
2
1.5
1
0.5
0
−0.5
0.5
1
1.5
2
2.5

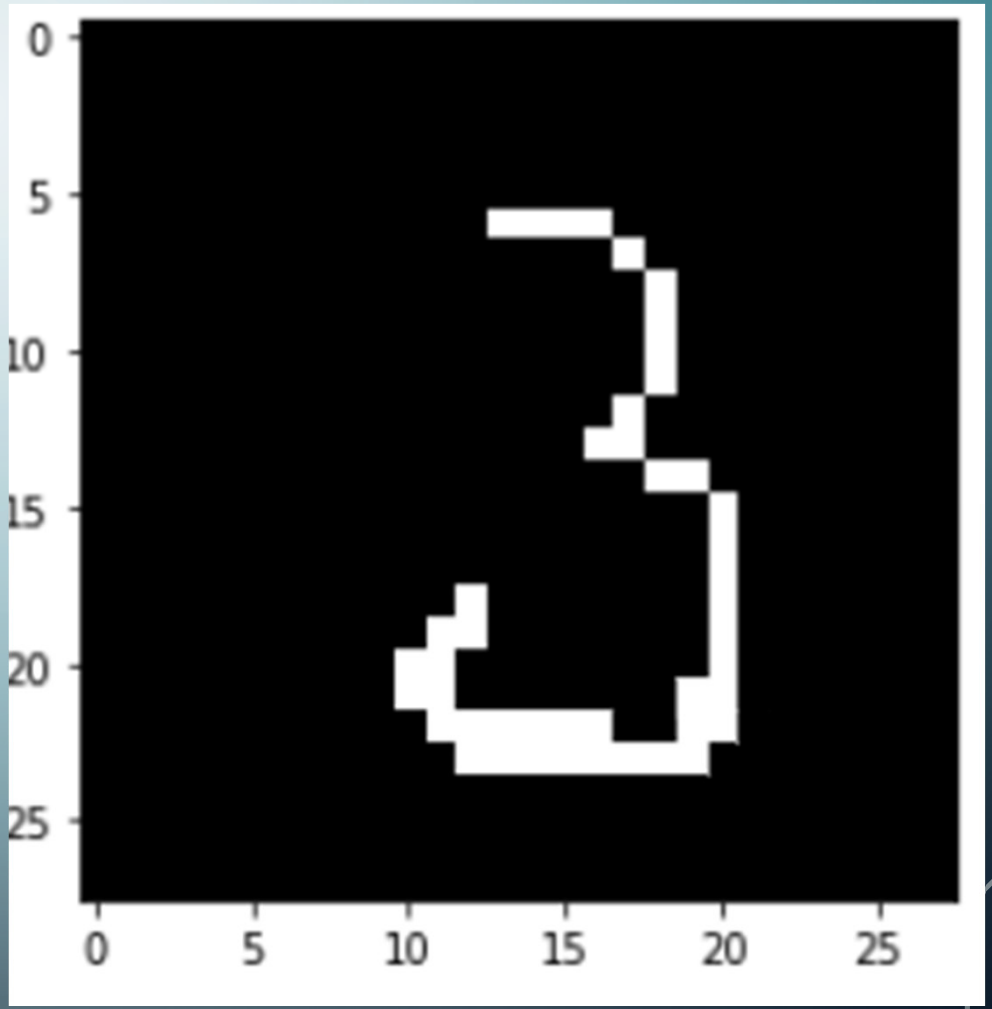

0
5
10
15
20
25
0
5
10
15
20
25

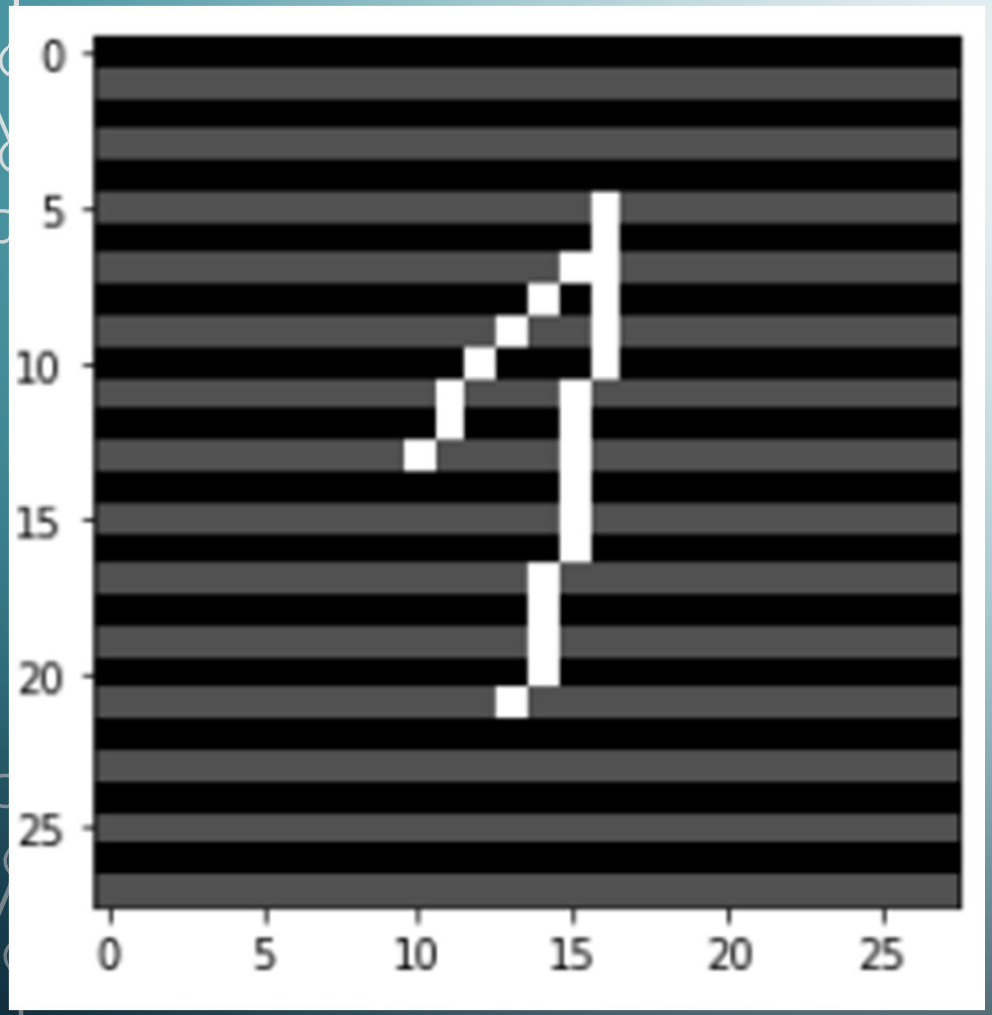

0
5
10
15
20
25
0
5
10
15
20
25

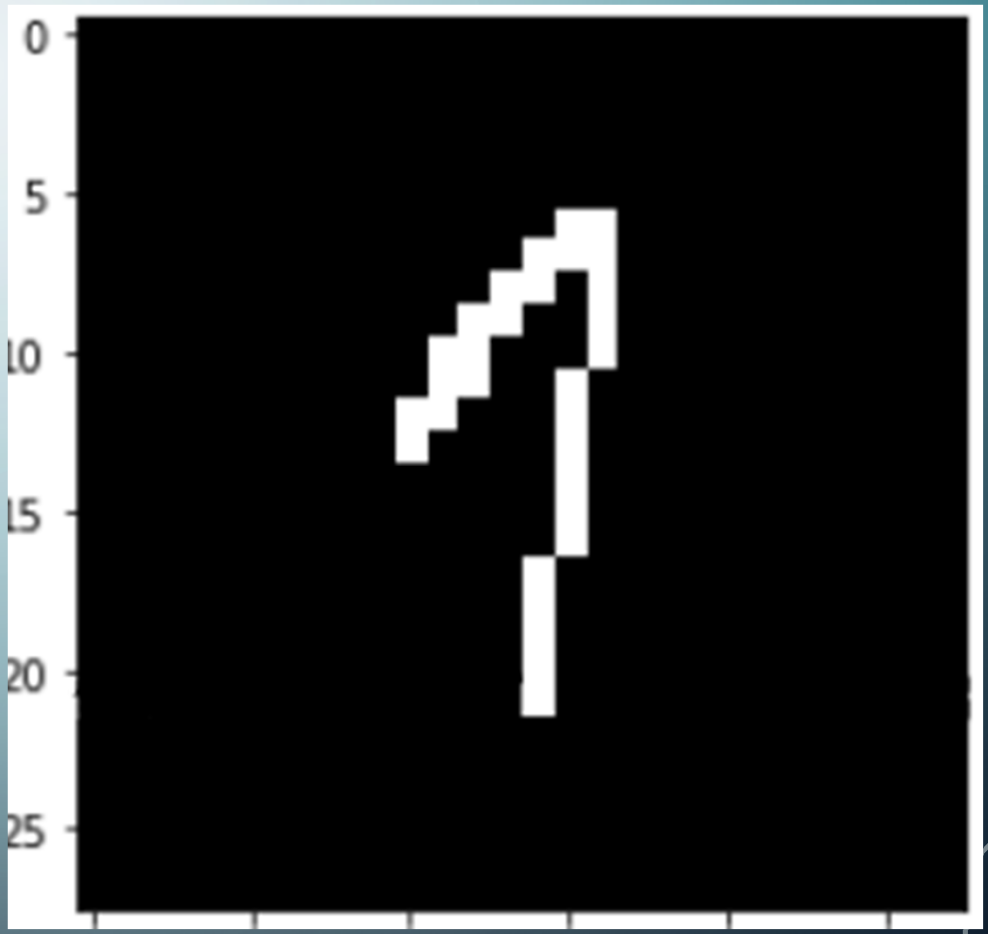
```
if matrix[mx,my] < 0.5:
   break
i =i+ 1
```

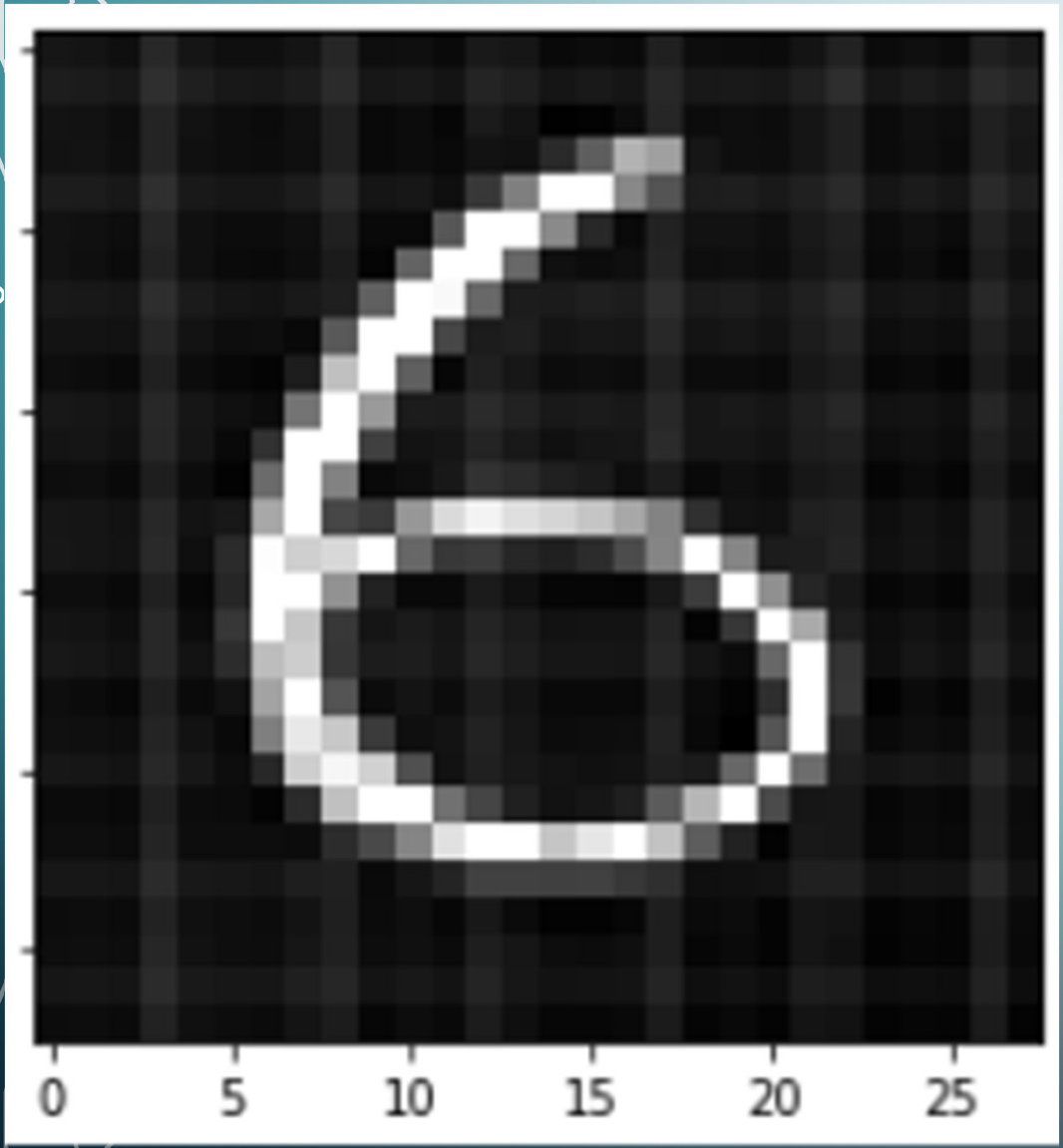

0
5
10
15
20
25
0
5
10
15
20
25

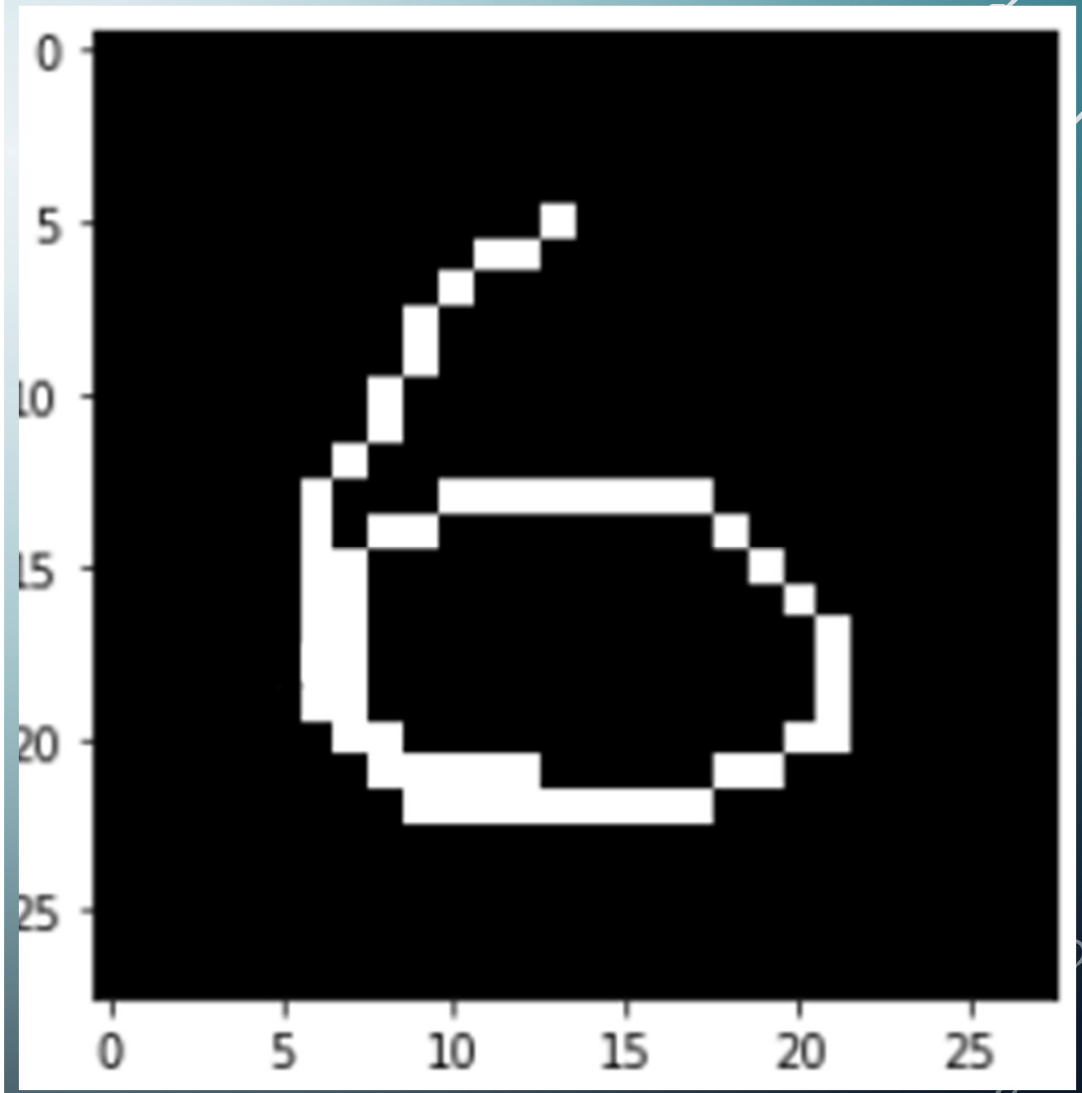

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25


0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


~45%
Erfolgsquote

Note	Prozent	verbal	Note	Grade	in words
1.0	≥ 95	sehr gut	1.0	A	excellent
1.3	≥ 90	sehr gut	1.3 - 1.5	A	excellent
1.7	≥ 85	gut	1.6 - 1.7	B	very good
2.0	≥ 80	gut	2.0	B	very good
2.3	≥ 75	gut	2.1 - 2.3	C	good
2.7	≥ 70	befriedigend	2.7 - 2.9	C	good
3.0	≥ 65	befriedigend	3.0	D	satisfactory
3.3	≥ 60	befriedigend	3.3 - 3.5	D	satisfactory
3.7	≥ 55	ausreichend	3.6 - 3.7	E	sufficient
4.0	≥ 50	ausreichend	4.0	E	sufficient
5.0	≤ 50	nicht ausreichend	4.1 - 5.0	FX and F	fail