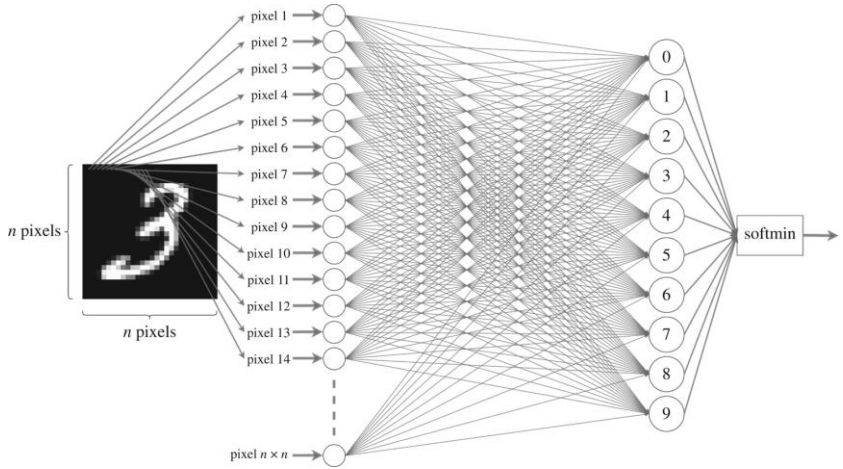
Optimal Output Layer Configuration of Artificial Neural Networks for Multiclass Classification

Dmitrii Bakhitov and Professor Sung-Hyuk Cha

How does the output layer look?



$$\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} = 0 \quad \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} = 1 \quad \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} = 2 \quad \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} = 3 \quad \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} = 4 \quad \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix} = 5 \quad \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix} = 6 \quad \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} = 7$$

Fact: number of output layer units is equal to number of classes.

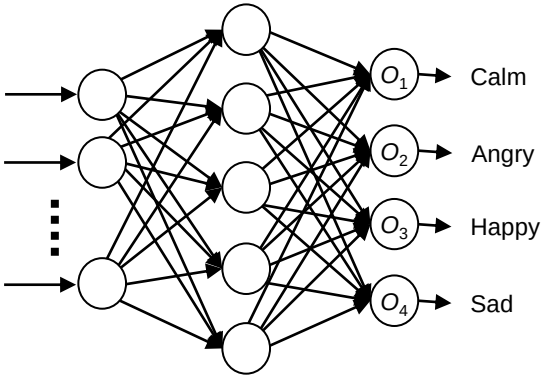
Why: The target feature is categorical and usually we use One Hot Encoding.

Result: Each output neuron becomes responsible for one particular class.

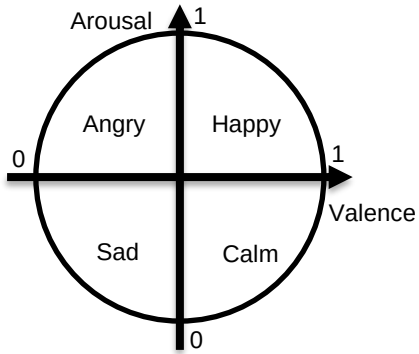
Question: Could we build an accurate and reliable neuron network model using less units on the output layer?

Answer: Yes

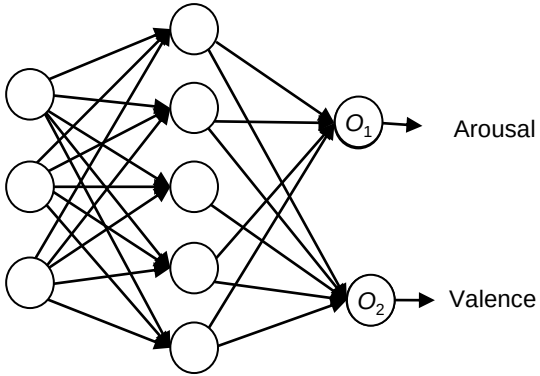
Example of using output layer units decimation



Neural Network with 4 output neurons



Russell's circumplex model



Neural Network with 2 output neurons

Calm : (1,0,0,0)
Angry : (0,1,0,0)
Happy : (0,0,1,0)
Sad : (0,0,0,1)

Russell's circumplex model

	Arousal	Valence
Calm	0	1
Angry	1	0
Happy	1	1
Sad	0	0

More Than One Hot Encoding

How we can encode N classes by less than N binary digits?

Classic way MNIST example

0 : (1,0,0,0,0,0,0,0,0,0)
1 : (0,1,0,0,0,0,0,0,0,0)
2 : (0,0,1,0,0,0,0,0,0,0)
3 : (0,0,0,1,0,0,0,0,0,0)
4 : (0,0,0,0,1,0,0,0,0,0)
5 : (0,0,0,0,0,1,0,0,0,0)
6 : (0,0,0,0,0,0,1,0,0,0)
7 : (0,0,0,0,0,0,0,1,0,0)
8 : (0,0,0,0,0,0,0,0,1,0)
9 : (0,0,0,0,0,0,0,0,0,1)

10 Classes

10 units

Decimating

Not reliable

0 :(0,0,1,0)
1 :(1,0,0,0)
2 :(0,0,1,1)
3 :(0,1,0,0)
4 :(0,1,0,1)
5 :(0,1,1,0)
6 :(1,1,0,0)
7 :(0,0,0,1)
8 :(0,1,1,1)
9 :(1,1,0,1)

4 units

Reliable

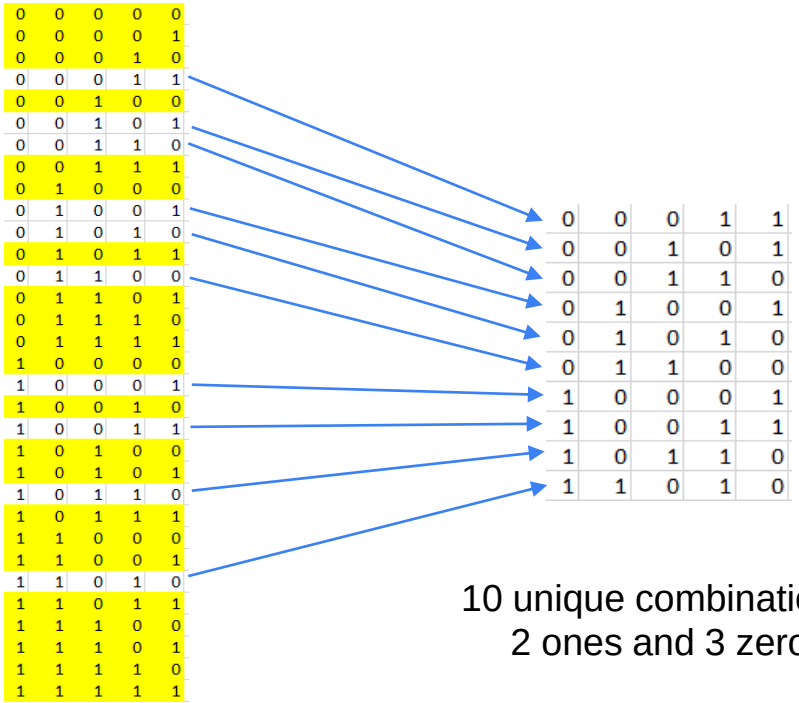
0:(0,0,0,1,1)
1:(0,0,1,0,1)
2:(0,0,1,1,0)
3:(0,1,0,0,1)
4:(0,1,0,1,0)
5:(0,1,1,0,0)
6:(1,0,0,0,1)
7:(1,0,0,1,0)
8:(1,0,1,0,0)
9:(1,1,0,0,0)

5 units

There are only 10 combination of

- 2 ones and
- 3 zeros

Decimating to 5 neuron output



0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1

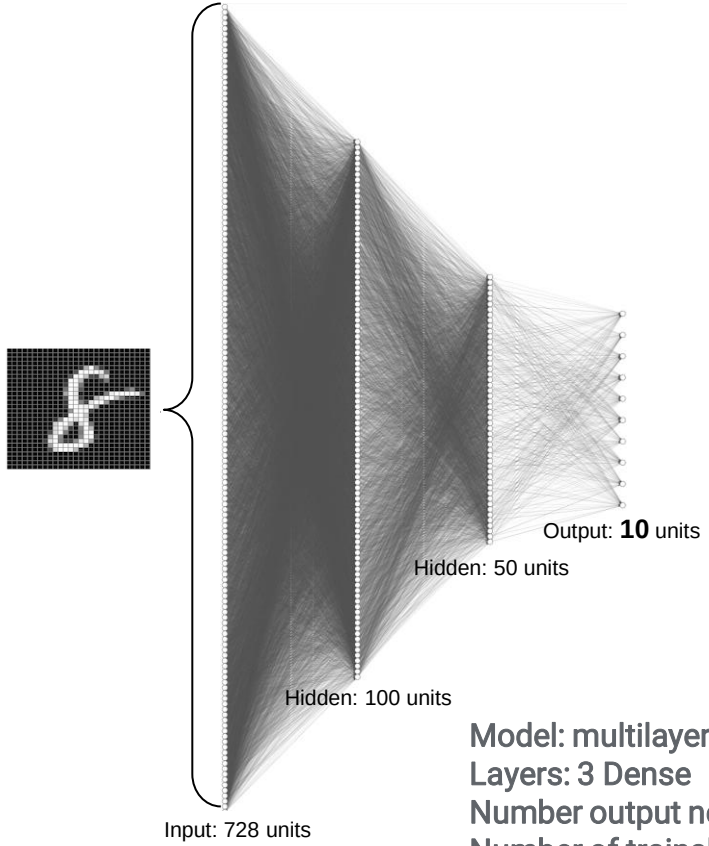
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	0

10 unique combination of
2 ones and 3 zeros

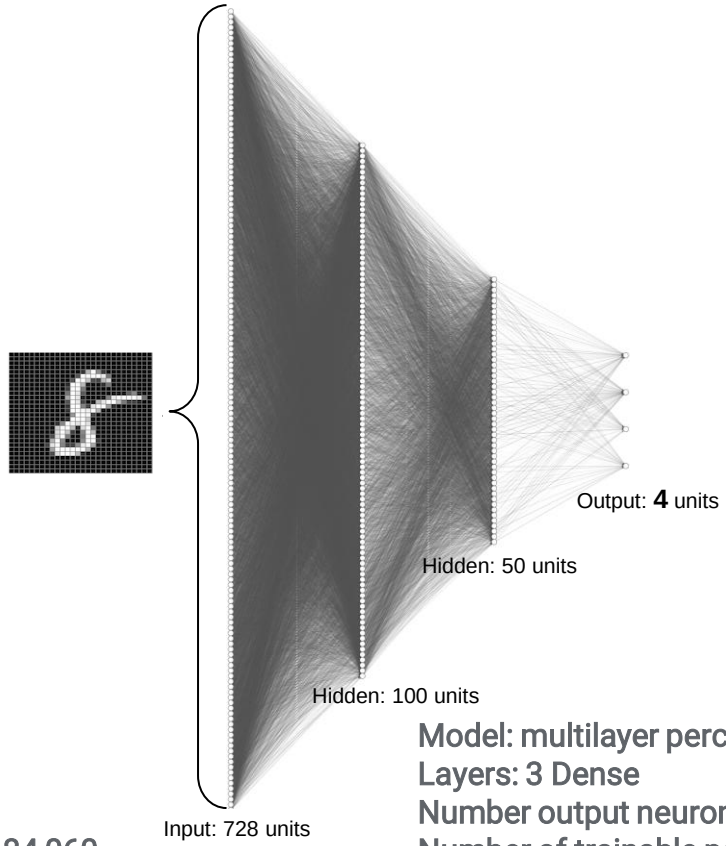
$$A(10\text{x}5) = \begin{vmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

32 possible combination
of 5 binary digits

Benefits of output neurons decimating



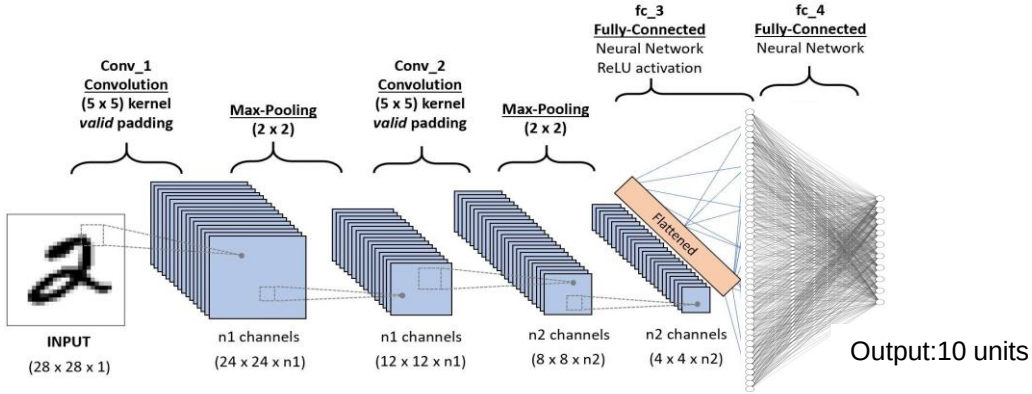
Model: multilayer perceptron
Layers: 3 Dense
Number output neurons : 10
Number of trainable parameters: 84,060



Model: multilayer perceptron
Layers: 3 Dense
Number output neurons : 4
Number of trainable parameters: 83,754

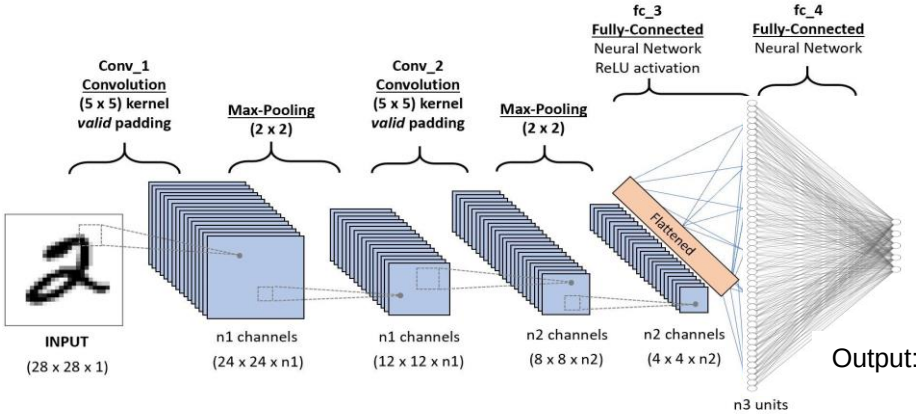
Benefits of output neurons decimating

Taking out 5 units provides a 30% complexity reduction!



Output:10 units

Model: CNN
Layers: 2 conv, 1 Dense
Number output neurons: 10
Number of trainable parameters: 34,826



Output:5 units

Model: CNN
Layers: 2 conv, 1 Dense
Number output neurons : 5
Number of trainable parameters: 26,821

Encoding

How encode a categorical feature of 10 categories into 5 units binary sequence?

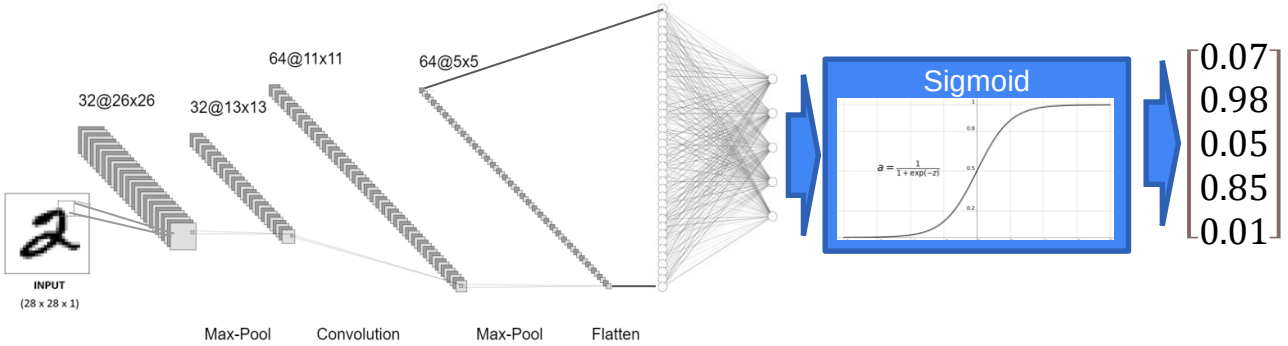
- Build a matrix A(10x5) where each of 10 rows is a binary sequence of 2 ones and 3 zeros
- Perform one hot encoding for target feature **y**
- Map encoded target feature by dot product into binary sequence: $y^T \cdot A$ = y'

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$y^T \cdot A = [00110]$$

Implementation

```
model_5 = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(5, activation="sigmoid"),
    ]
)
```




```
dict5 = {(0,0,0,1,1):0,
         (0,0,1,0,1):1,
         (0,0,1,1,0):2,
         (0,1,0,0,1):3,
         (0,1,0,1,0):4,
         (0,1,1,0,0):5,
         (1,0,0,0,1):6,
         (1,0,0,1,0):7,
         (1,0,1,0,0):8,
         (1,1,0,0,0):9}
```

$$\text{DoubleMaxRounding}\left(\begin{bmatrix}0\\0.9\\0\\0.8\\0.1\end{bmatrix}\right)=\begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix}$$

$$\text{DoubleMaxRounding}\left(\begin{bmatrix}0.1\\0.7\\0\\0.4\\0\end{bmatrix}\right)=\begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix}$$

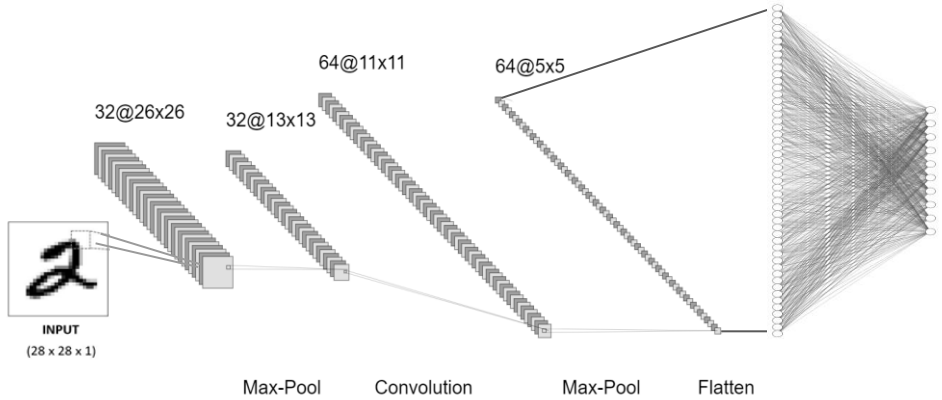
- Output layer requires activation function
- SoftMax is not suitable for our model
- Sigmoid function is good solution for this model but not enough
- A DoubleMaxRounding function is required
- We have to round up maximum value and second order maximum value, the rest values should be rounded down
- After DoubleMaxRounding output vector could be mapped into 10 categories by simple dictionary

Results

Baseline model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dropout (Dropout)	(None, 1600)	0
dense (Dense)	(None, 10)	16010

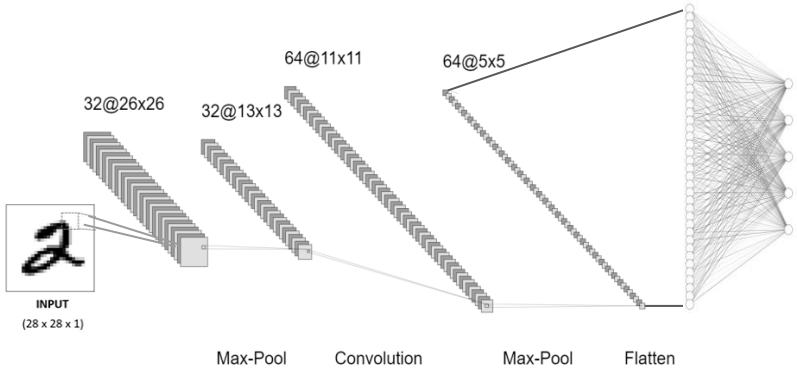
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0



Number params = 34 826
Number Conv layers = 2
Accuracy = 98.9%

Results

5 neurons model

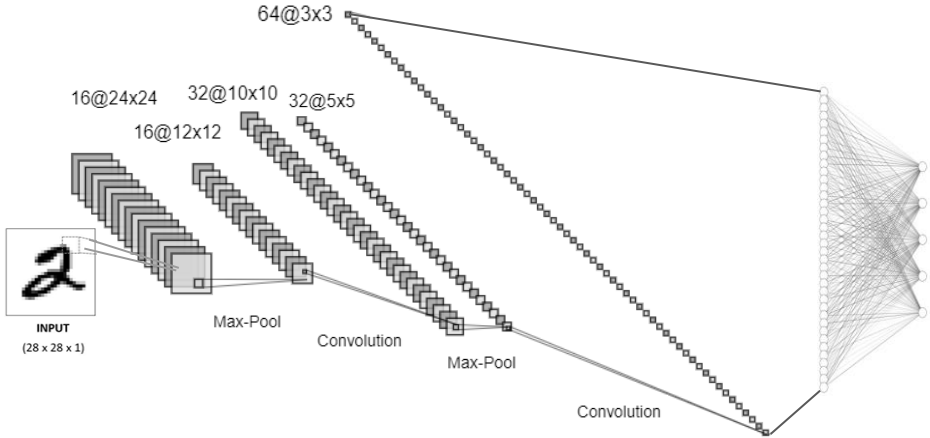


Number params = 26 821
Number Conv layers = 2
Accuracy = 97.4%

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_4 (MaxPooling 2D)	(None, 13, 13, 32)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_5 (MaxPooling 2D)	(None, 5, 5, 64)	0
flatten_2 (Flatten)	(None, 1600)	0
dropout_2 (Dropout)	(None, 1600)	0
dense_2 (Dense)	(None, 5)	8005

Total params: 26,821
Trainable params: 26,821
Non-trainable params: 0

5 + 1 conv neurons model



Number params = 26 821
Number Conv layers = 3
Accuracy = 99.2%

Layer (type)	Output Shape	Param #
conv2d_292 (Conv2D)	(None, 24, 24, 16)	416
max_pooling2d_243 (MaxPooli ng2D)	(None, 12, 12, 16)	0
conv2d_293 (Conv2D)	(None, 10, 10, 32)	4640
max_pooling2d_244 (MaxPooli ng2D)	(None, 5, 5, 32)	0
conv2d_294 (Conv2D)	(None, 3, 3, 64)	18496
flatten_111 (Flatten)	(None, 576)	0
dropout_110 (Dropout)	(None, 576)	0
dense_111 (Dense)	(None, 5)	2885

Total params: 26,437
Trainable params: 26,437
Non-trainable params: 0

Conclusion

	Baseline (10 neurons)	4 neurons	4 neurons +1 Conv	5 neurons	5 neurons +1 Conv
Number of trainable parameters	34,826	25,220	25,860	26,821	26,437
Number of Convolutional layers	2	2	3	2	3
Accuracy, %	98.9	95.9	98.1	97.9	99.2

Our method allowed us to reduce the trainable parameters from 34,826 to 26,437 including one additional convolutional layer with no decoding errors and a 99.2% accuracy rate.

Further steps:

- Check if it matters how digits are assigned to binary vectors.
- If the accuracy depends on how we assign digits to vectors we will try to interpret this dependence