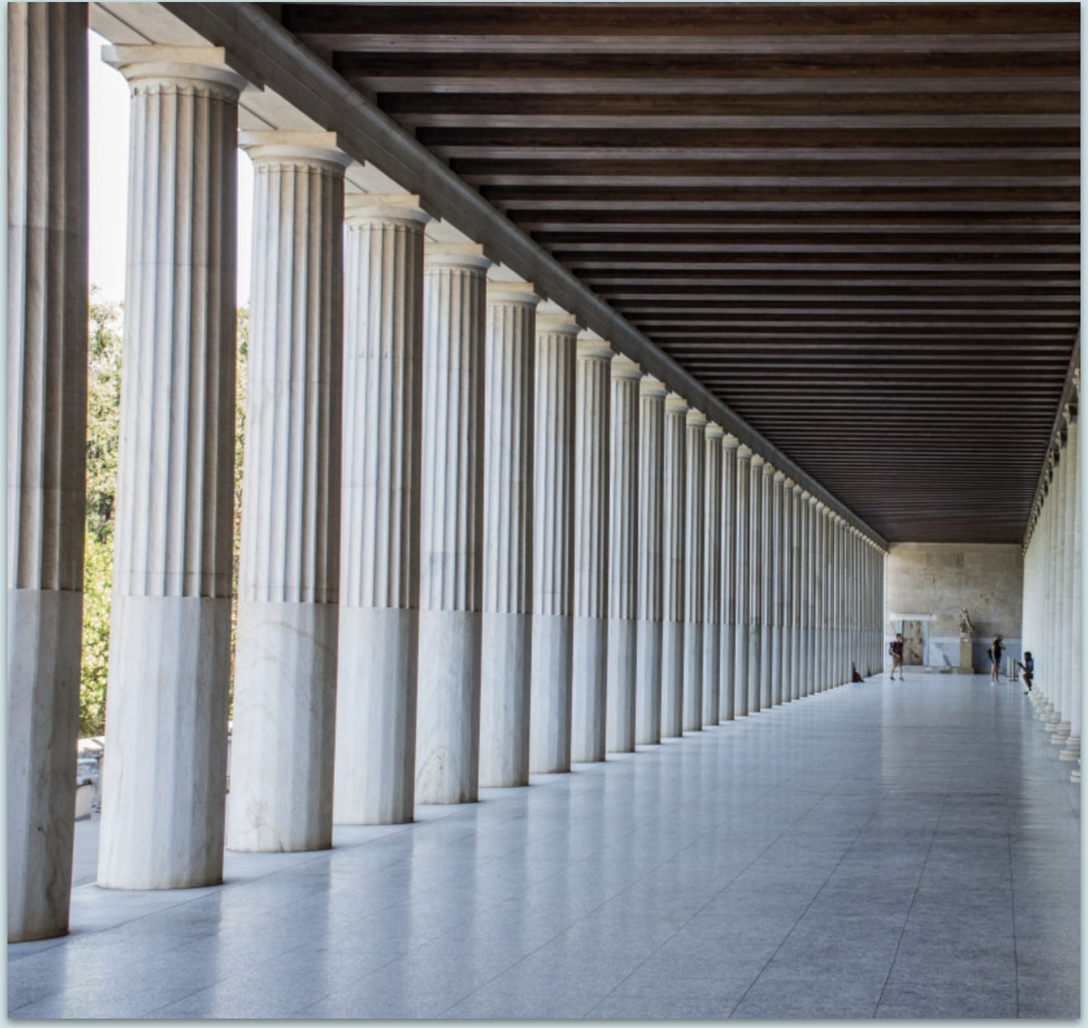
Going Deeper with CNN

Study of Classical Architectures

ivanovitch.silva@ufrn.br
@ivanovitchm

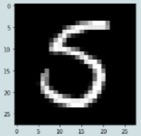
UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DCA



Deeper Layers

LeNet



1998

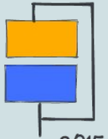
AlexNet

2012

VGG

Components
and groups
2014

ResNet



2015

Research understood

+ accuracy
+ layers

+ weight vanishing
+ weight exploding

Wide CNN

Inception V1
2014

Inception V2, V3
2015

ResNext
2015

New Tools

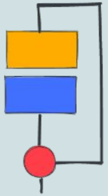
Batch Normalization
(improve weights vanish, explod.)

Dropout
(attacked the memorization in
deer layers)

Overfitting problems persist

Connectivity Patterns

Residual Block (RB)



DenseNet
2017

Xception
2017

SE-Net
2017

reduce memorization
without impact to much in complexity
Improve RB connectivity
or between RB

models without memory restriction

Mobile CNN

MobileNet V1
2017

MobileNet V2
2018

Shuffle Net
2017

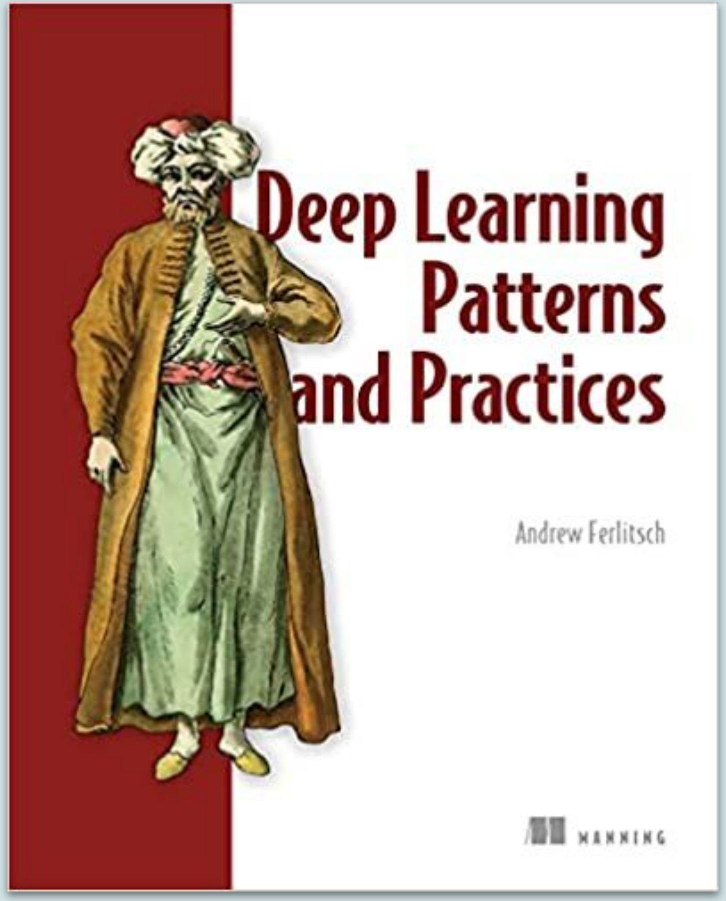
TensorFlow Lite
2017 >> 2019

Compact models
Accuracy vs latency

Compression and
Quantization

Refactoring of
Convolution

Accuracy can be
compromised



Part 2: Basic Design Patterns

Chapters 5 to 8

PROC. OF THE IEEE, NOVEMBER 1998 1

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract—

Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.

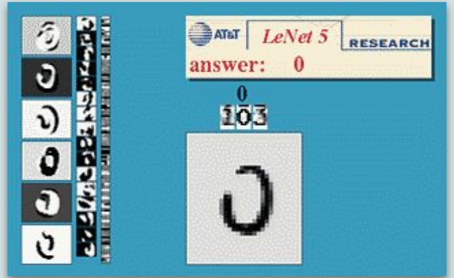
Keywords— **Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.**

I. Introduction

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

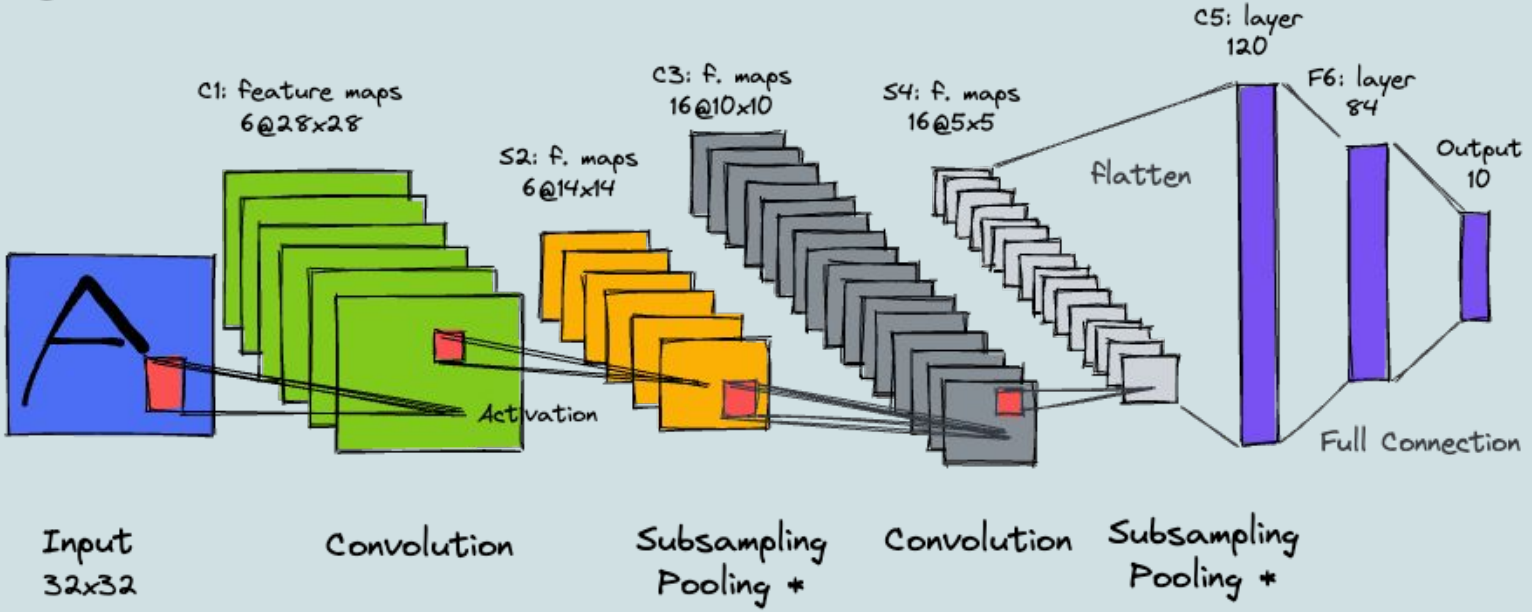
The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it almost impossible to build an accurate recognition system



http://yann.lecun.com/exdb/lenet/

LeNet-5



INPUT => [[CONV => RELU] * N => POOL?] * M => [FC => RELU] * K => FC

Fixed-sized input images.

Group convolutional and pooling layers into blocks.

Repetition of convolutional-pooling blocks in the architecture.

Increase in the number of filters with the depth of the network.

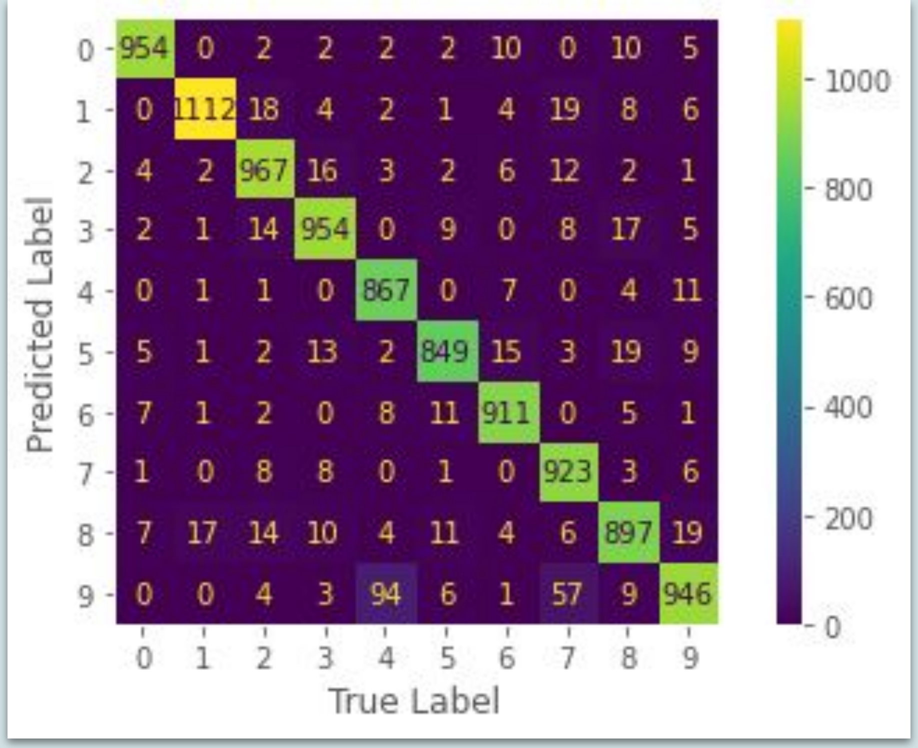
Distinct feature extraction and classifier parts of the architecture.


Training Loss and Accuracy
Loss/Accuracy
Epoch #
train_loss
val_loss
train_acc
val_acc
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

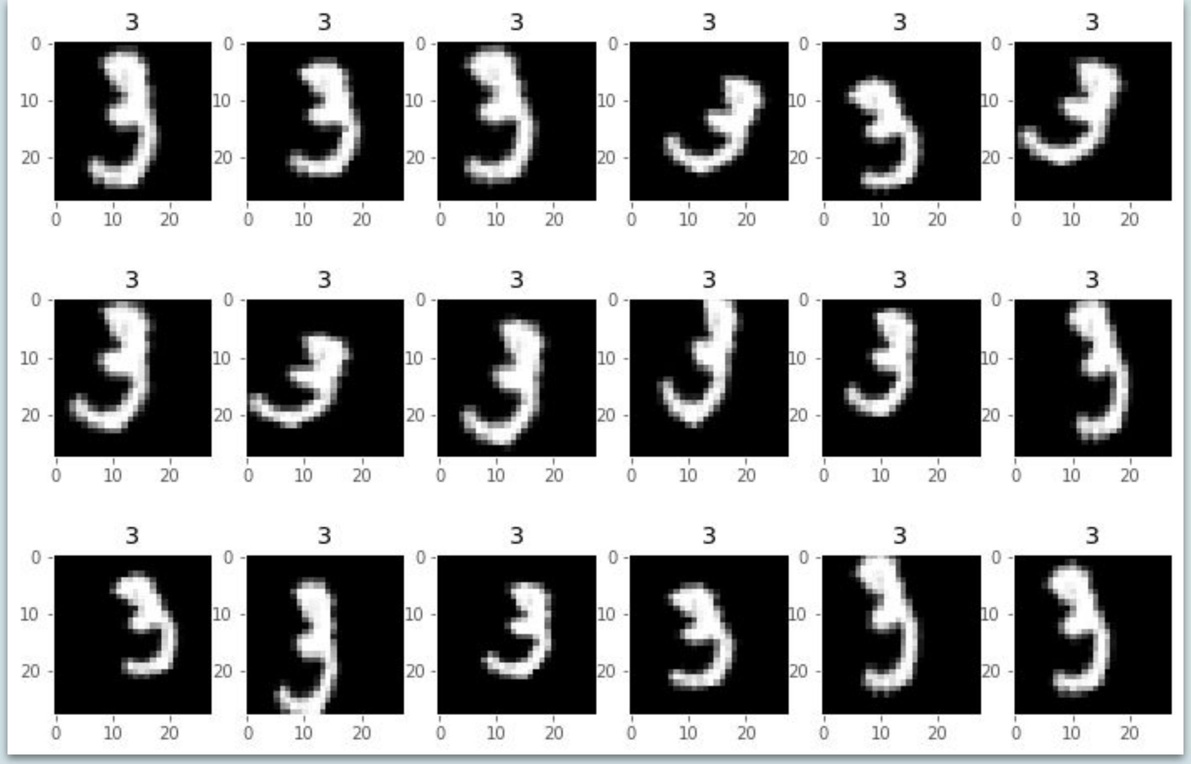
```
313/313 [==============================] - 1s 2ms/step
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       980
           1       0.95      0.98      0.96      1135
           2       0.95      0.94      0.94      1032
           3       0.94      0.94      0.94      1010
           4       0.97      0.88      0.93       982
           5       0.92      0.95      0.94       892
           6       0.96      0.95      0.96       958
           7       0.97      0.90      0.93      1028
           8       0.91      0.92      0.91       974
           9       0.84      0.94      0.89      1009

    accuracy                           0.94     10000
   macro avg       0.94      0.94      0.94     10000
weighted avg       0.94      0.94      0.94     10000
```



Data Augmentation




Training Loss and Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Loss/Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch #
train_loss
val_loss
train_acc
val_acc

Without Data Augmentation

313/313 [==============================] - 1s 2ms/step

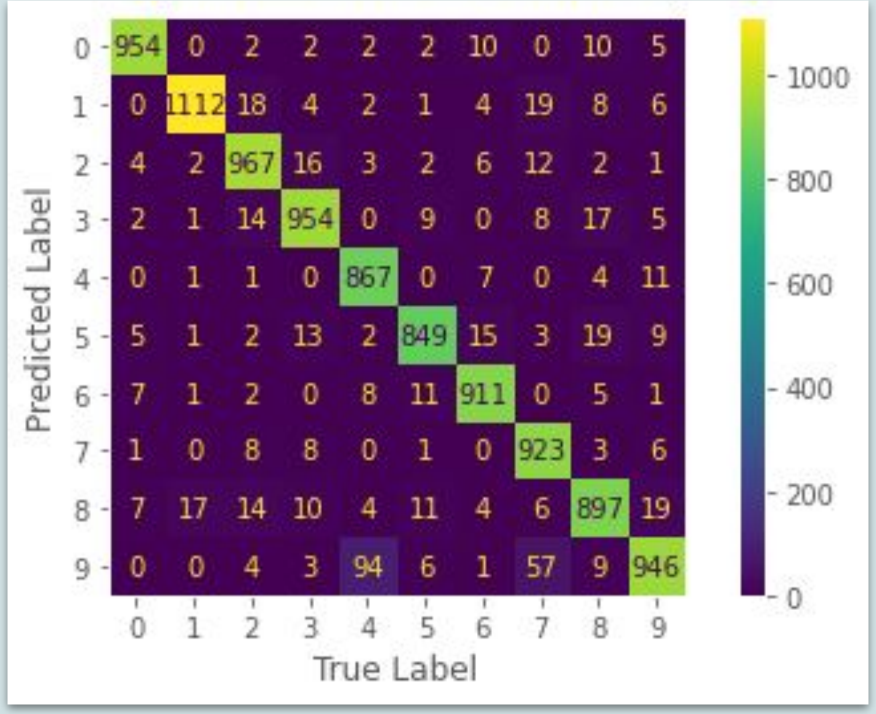
	precision	recall	f1-score	support
0	0.97	0.97	0.97	980
1	0.95	0.98	0.96	1135
2	0.95	0.94	0.94	1032
3	0.94	0.94	0.94	1010
4	0.97	0.88	0.93	982
5	0.92	0.95	0.94	892
6	0.96	0.95	0.96	958
7	0.97	0.90	0.93	1028
8	0.91	0.92	0.91	974
9	0.84	0.94	0.89	1009
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

With Data Augmentation

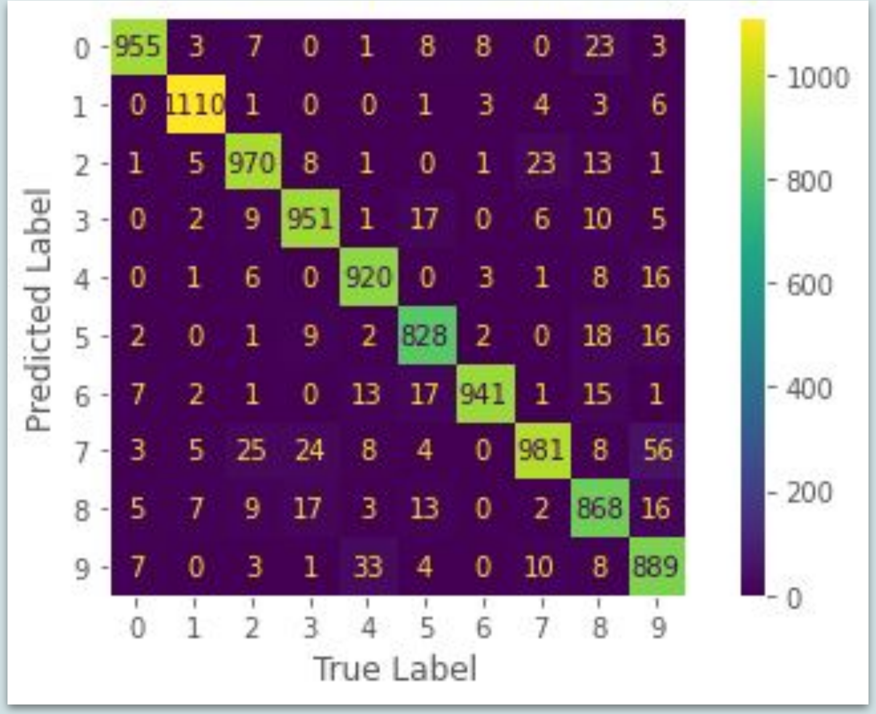
313/313 [==============================] - 1s 2ms/step

	precision	recall	f1-score	support
0	0.95	0.97	0.96	980
1	0.98	0.98	0.98	1135
2	0.95	0.94	0.94	1032
3	0.95	0.94	0.95	1010
4	0.96	0.94	0.95	982
5	0.94	0.93	0.94	892
6	0.94	0.98	0.96	958
7	0.88	0.95	0.92	1028
8	0.92	0.89	0.91	974
9	0.93	0.88	0.91	1009
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

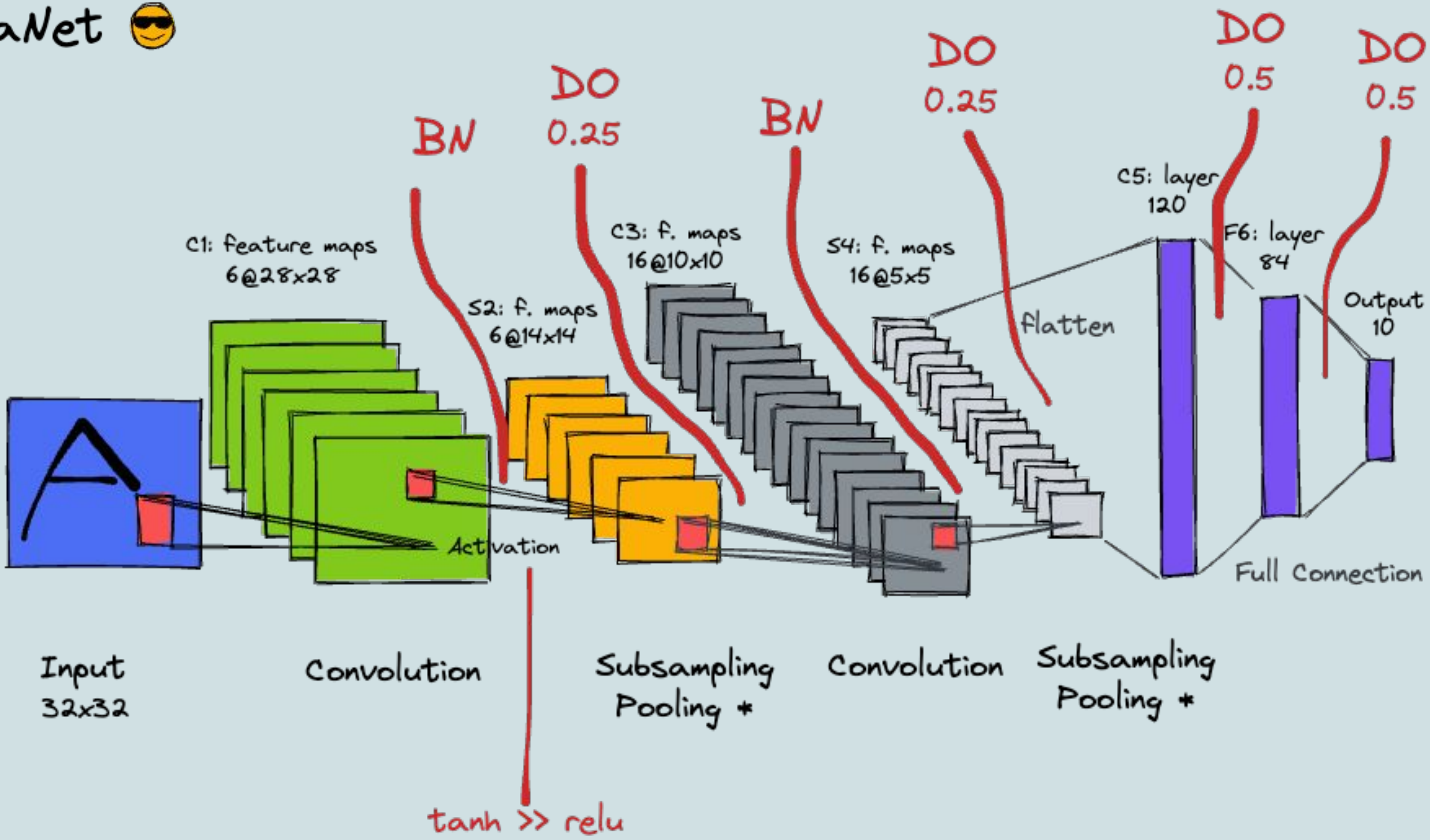
Without
Data Augmentation



With
Data Augmentation



IvaNet 😎


BN
DO
0.25
BN
DO
0.25
DO
0.5
DO
0.5
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps
16@10x10
S4: f. maps
16@5x5
flatten
C5: layer
120
F6: layer
84
Output
10
Activation
Full Connection
Input
32x32
Convolution
Subsampling
Pooling *
Convolution
Subsampling
Pooling *
tanh >> relu