LAUSHON NEFERKARA

2016-10-19

GENERAL ASSEMBLY DATA SCIENCE COURSE PROJECT

KAGGLE CHALLENGE:

FACIAL KEYPOINTS DETECTION

THE COMPETITION

THE COMPETITION

The objective is given a set of face images, predict 15 keypoint positions.

THE COMPETITION

Predict:

1. left_eye_center
2. right_eye_center
3. left_eye_inner_corner
4. left_eye_outer_corner
5. right_eye_inner_corner
6. right_eye_outer_corner
7. left_eyebrow_inner_end
8. left_eyebrow_outer_end
9. right_eyebrow_inner_end
10. right_eyebrow_outer_end
11. nose_tip
12. mouth_left_corner
13. mouth_right_corner
14. mouth_center_top_lip
15. mouth_center_bottom_lip

THE COMPETITION

- Pictures are black-and-white, 96 x 96 pixels.
- Each of the 15 keypoints has an (x,y) coordinate.
- Regression problem where the target is a vector of 30 (15x2) values bounded between 0 and 96, the size of the image.
- Entries evaluated on Root Mean Squared Error (RMSE)

WHY?

- Used as a building block in several applications
  - tracking faces in images and video
  - analyzing facial expressions
  - detecting dysmorphic facial signs for medical diagnosis
  - biometrics / face recognition

WHY?

- Very challenging problem. Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination.
- Personal: The popular solution for the challenge is Neural Networks. And, I wanted to gain a better understanding of NNs.

THE DATA

THE DATA

- Kaggle provides 2 datasets:
  - training.csv
  - test.csv which I will call the “competition data” (so as to not confuse it with test data split from the training set)
- Problem: Missing data
  - Number of rows: 7049
  - All keypoints: 2140

THE DATA

THE SOLUTION

DATA ENGINEERING

DATA ENGINEERING

- Missing keypoints –> small dataset
- NNs are prone to overfitting. (Low bias, high variance)
  - An overfitting net can generally be made to perform better by using more training data.
- Need more data

DATA AUGMENTATION

- Lets us artificially increase the number of training examples by applying transformations, adding noise etc.
- More economical than having to go out and collect more examples by hand.
- Mirrored existing data (images & keypoints)

THE MODEL

THE MODEL

- I looked at how others approached the solution
- Convolutional Neural Networks (CNN) was most popular
- So, I decided to try to use a CNN and see how far I could get

CONVOLUTIONAL NEURAL NETWORKS

ARE HARD.

THE END

GOT SOME HELP


FIGHTER?
FWAKK
SHOOT!
KITTY!
YAY! NOW IS FIGHTY TIME! FIGHTY TIME! BLOOD BLOOD BLOOD.
SHLKT
Hey, I knew I recognized that chump ...
... ain't that the guy what did the Four Winds job?
Go away, little nobody.
Mister, you'd best let this go, before I get peeved.
OVER MY... UH... COMPETITION.
You just killed the nice deranged chick from the juice bar that I was gonna score with someday maybe!
?!?
THE -- THE BLACK... HEH... HEH...
BWAAAH
CHICKEN HEAD! BWAHH!
THIS IS THE CEREMONIAL HEADDRESS OF THE ANCIENT LOA! IT IS AN HONOR TO --
AAAAHH HAAAA!
THE -- CHICKEN -- SAID -- PEEP! HAHAH!
IF YOU'RE QUITE FINISHED...
KILL HIM.
MASK CAME 6165.
OF MY EARLIEST ...TS -- KILLING THIS CYBORG MUTANT ...M THE FUTURE.
OH
HMMM... I DO.
I HAD THE SWORD OUT ALREADY AND, WELL, HE TOOK A LOT OF MY BRAIN. I FIGURED, IT'S JUST A LITTLE PANCREAS.
UHM, WHERE IS IT?
YOU KNOW HOW WE HAVE THESE LITTLE POUCHES ON OUR BELTS, BUT WE NEVER SEEM TO USE THEM...
DES, HE DOESN'T NEED IT! WE BOTH HEALING FACTORS OUR SYSTEMS!
HIS HAS PROBABLY GROWN BACK ALREADY! IT'S THE PRINCIPLE OF THE THING!
WHEN I FROST YOUR SORRY OLD MECHANICAL BUTT, DON'T TAKE IT PERSONALLY, OKAY?


DEADPOOL

SURPRISE!!!

SORRY I COULDN'T BE THERE.

AND, I ADDED MORE SLIDES!

P.S. DON'T SUCK

DEADPOOL


YO, NERDS!

AFTER A LOT OF HARD WORK


1001 jokes about Captain America

WHAT IS A CONVOLUTIONAL NEURAL NETWORK?


100
MG
DEADPOOL

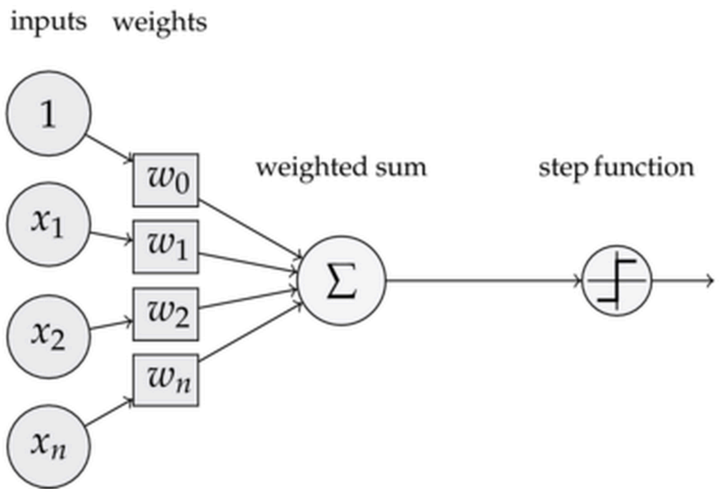
WHAT IS A CONVOLUTIONAL NEURAL NETWORK?

- Convolutional Neural Networks (ConvNets, CNNS) are a special class of Neural Networks.
- To explain what they are, let's first make a quick review of what we learned in class.

NEURAL NETWORKS

- Inputs
- Weights
- Activation function
- Backpropagation
- Fully-connected neural network
- Unsupervised feature learning

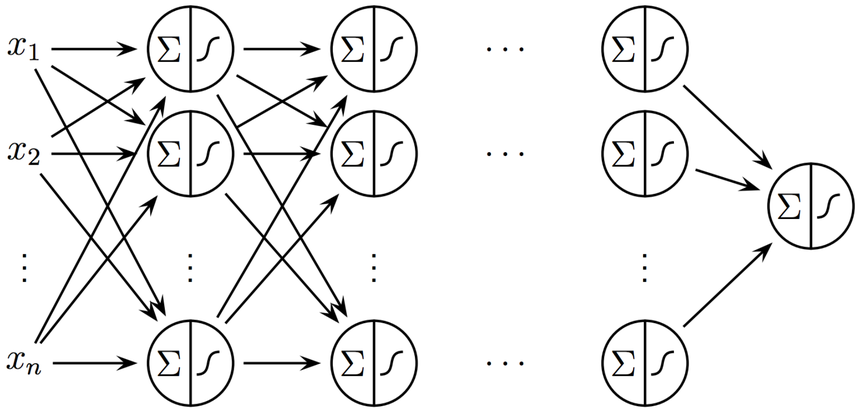
NERUAL NETWORKS



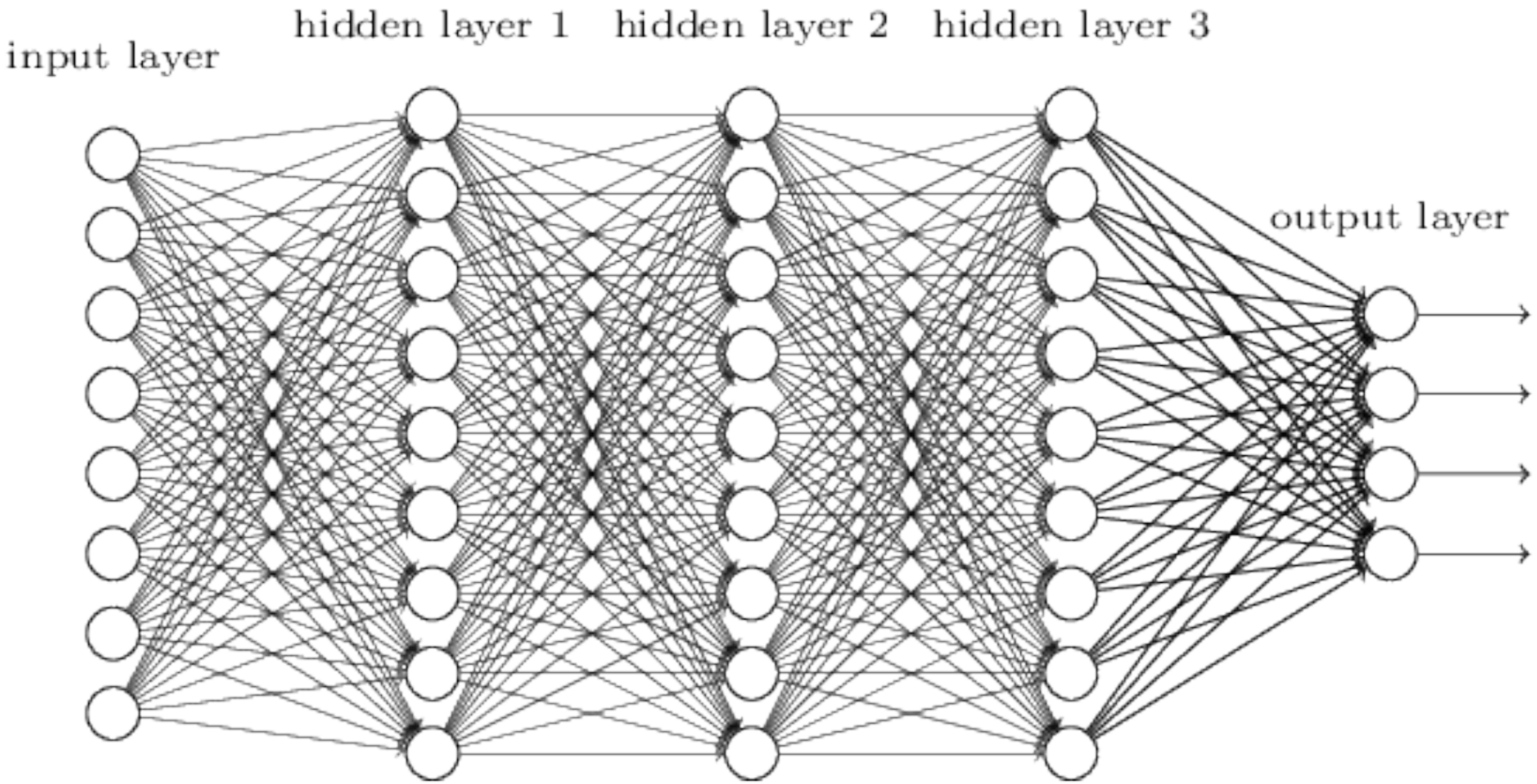
$$f_{log}(z) = \frac{}{1 + e^{-z}}$$

f_{log} is called logistic function

A multi layer perceptrons (MLP) is a finite acyclic graph. The nodes are neurons with logistic activation.



FULLY-CONNECTED NEURAL NETWORK



FULLY CONNECTED NEURAL NETWORK

- Network architecture does not take into account the spatial structure of the images
- It treats input pixels which are far apart and close together on exactly the same footing
- Spatial structure must instead be inferred from the training data

But what if, instead of starting with a network architecture which is a blank slate, we used an architecture which tries to take advantage of the spatial structure?


BLAM!

CONVOLUTIONAL NEURAL NETWORK

WHY A CONVOLUTIONAL NEURAL NETWORK?

- Use a special architecture which is particularly well-adapted to analyze images.
- This architecture makes the network fast to train
- Being fast helps us train deep, many-layer networks
- Today, deep convolutional networks are used in most neural networks for image recognition

CONVOLUTIONAL NEURAL NETWORKS

- Recognize visual patterns directly from pixel images with minimal preprocessing.
- They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.
- Work very well for analyzing any spatial data
- If the data can be made to look like an image, CNNs are useful. But, if your data is just as useful after swapping any of the columns with each other, then CNNs would not be a good solution.

INSPIRATION

- CNNs are biologically-inspired. From research on cats, we know the visual cortex contains a complex arrangement of cells.
- These cells are sensitive to small sub-regions of the visual field, called a receptive field. The sub-regions are tiled to cover the entire visual field. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images.

INSPIRATION

- Two basic cell types have been identified:
  - Simple cells respond to specific edge-like patterns within their receptive field.
  - Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

CNN LAYERS

- If you feed them a bunch of pictures of faces for instance, they will learn some basic things like edges, dots, bright spots, dark spots in the first layer.
- In the second layer, we detect recognizable items such as eyes, noses, and mouths.
- In the third layer, are things that look like faces.

LeNet5

- LeNet5-style convolutional neural networks are at the heart of deep learning's recent breakthrough in computer vision.
- Developed by Yann LeCun, a leader in deep learning research
- Convolutional layers are different to fully connected layers; they use a few **techniques** to reduce the number of parameters that need to be learned, while retaining the important information of the data (spatial relations).

LENET5 CNNS TECHNIQUES

- **local connectivity:** neurons are connected only to a subset of neurons in the previous layer
- **weight sharing:** weights are shared between a subset of neurons in the convolutional layer (these neurons form what's called a feature map)
- **pooling:** static subsampling of inputs.


MADNESS.

COMPONENTS

COMPONENTS

- Convolutional neural networks use three basic ideas:
  - local receptive fields (convolution)
  - shared weights (convolution)
  - pooling
- Let's look at each of these ideas in turn.

LOCAL RECEPTIVE FIELDS

- In the fully-connected layers, the inputs are a vertical line of neurons.
- In a convolutional net, think of the inputs as a 96×96 square of neurons, whose values correspond to the 96×96 pixel intensities we're using as inputs.

CONVOLUTION

CONVOLUTION: LOCAL RECEPTIVE FIELDS

- As per usual, we'll connect the input pixels to a layer of hidden neurons.
- But we won't connect every input pixel to every hidden neuron. Instead, we only make connections in small, localized regions of the input image. Here I'm using a 3 x 3 region.

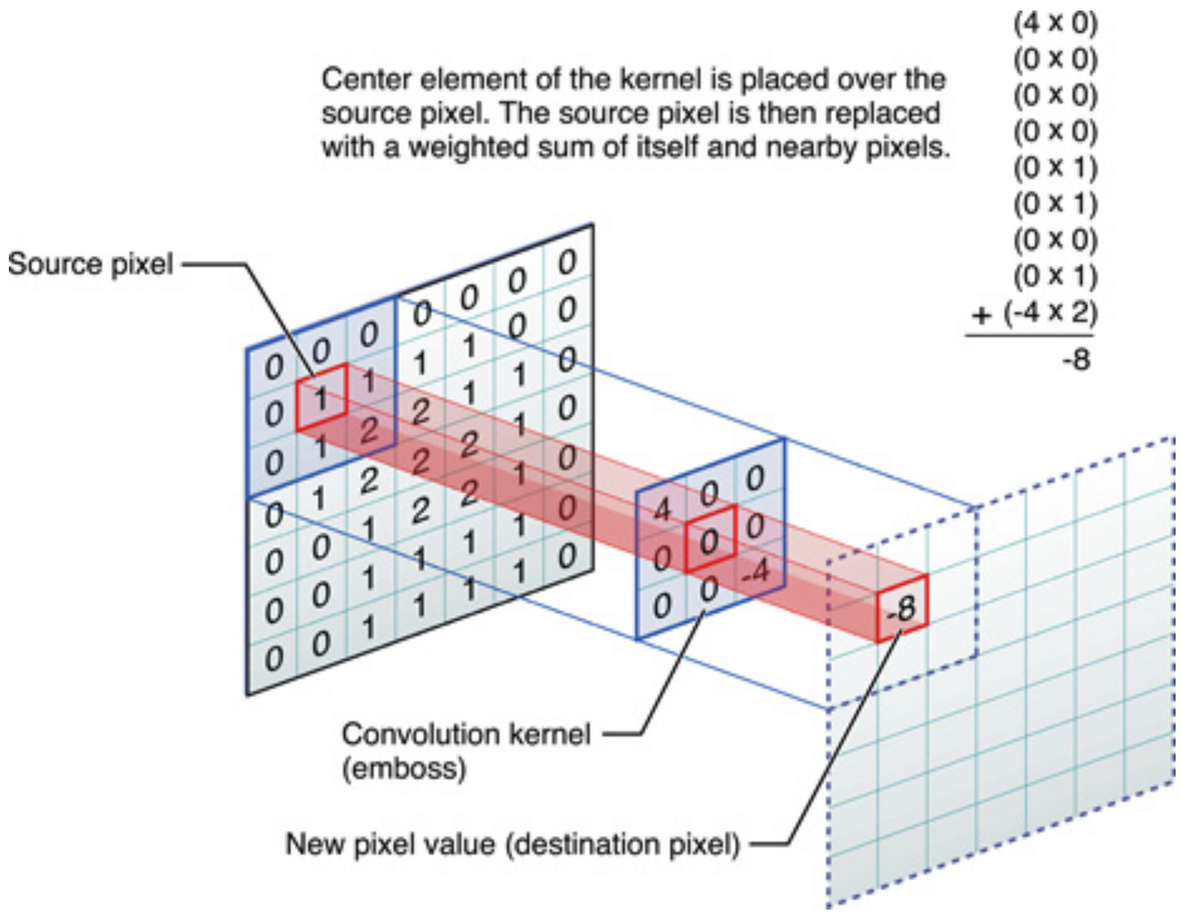
CONVOLUTION: LOCAL RECEPTIVE FIELDS

- That region in the input image is called the *local receptive field* for the hidden neuron. It's a little window on the input pixels.
- Each connection learns a weight. (And the hidden neuron learns an overall bias as well.)
- You can think of that particular hidden neuron as learning to analyze its particular local receptive field.

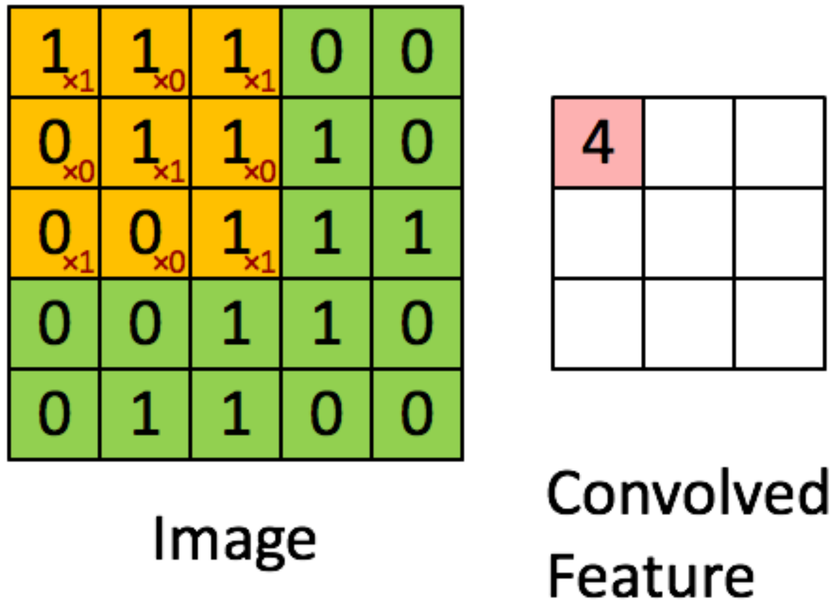
CONVOLUTION: LOCAL RECEPTIVE FIELDS

- We then “slide” the local receptive field across the entire input image.
- For each local receptive field, there is a different hidden neuron in the first hidden layer.
- 1st hidden layer will be 94 x 94 because of hitting the edges.

CONVOLUTION



CONVOLUTION



CONVOLUTION: SHARED WEIGHTS AND BIASES

- The CNN uses the same weights and bias for each of the hidden neurons.
- All the neurons in the first hidden layer detect exactly the same *feature*, just at different locations in the input image.
- Think of the feature detected by a hidden neuron as the kind of input pattern that will cause the neuron to activate: it might be an edge in the image, for instance, or maybe some other type of shape.

CONVOLUTION: SHARED WEIGHTS AND BIASES

- To see why this makes sense, suppose the weights and bias are such that the hidden neuron can pick out, say, a vertical edge in a particular local receptive field. That ability is also likely to be useful at other places in the image. And so it is useful to apply the same feature detector everywhere in the image.
- To put it in slightly more abstract terms, CNNs are well adapted to the translation invariance of images: move a picture of a cat (say) a little ways, and it's still an image of a cat.

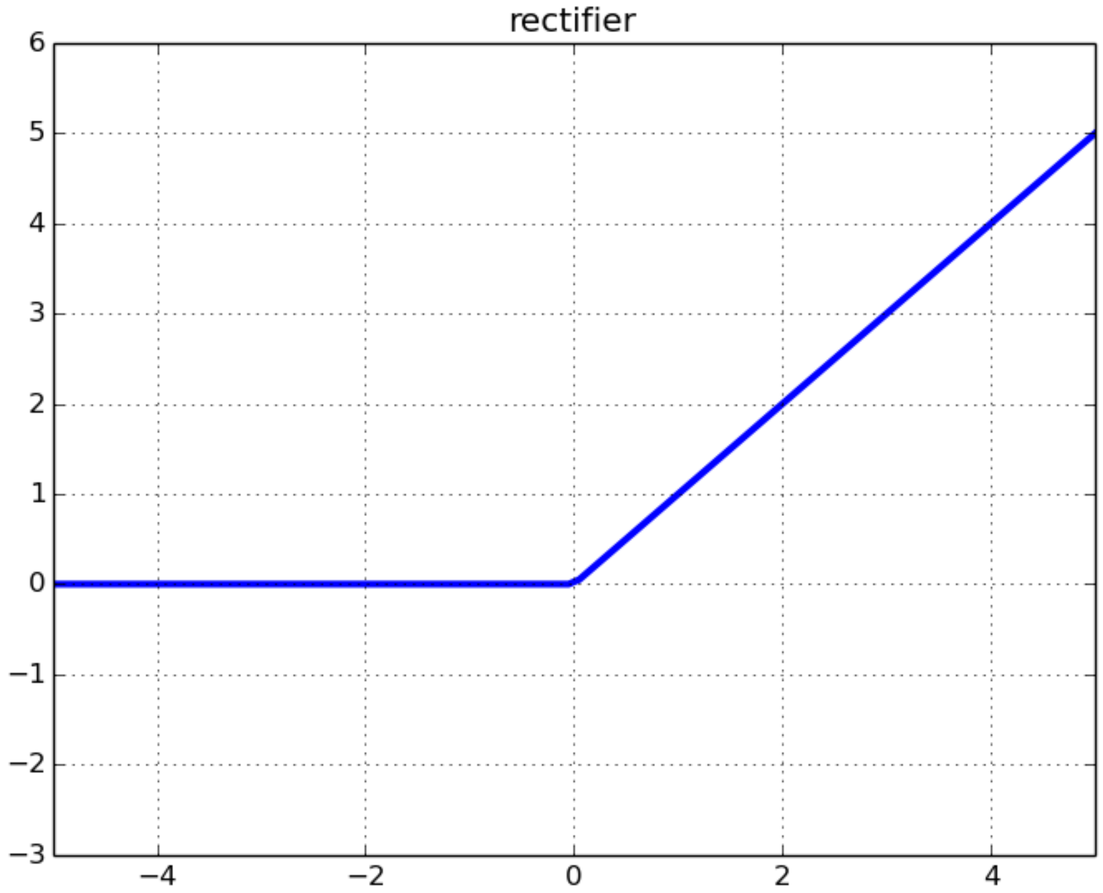
CONVOLUTION: SHARED WEIGHTS AND BIASES

- For this reason, we call the map from the input layer to the hidden layer a *feature map*.
- For image recognition we'll need several different feature maps. (My first layer has 32 features.)
- A big advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network.

CONVOLUTION: RELU (RECTIFIED LINEAR UNITS) LAYER

- At the end of the convolution layer, a ReLU activation layer is applied.
- Purpose: introduce nonlinearity to a system that basically has just been computing linear operations (just element-wise multiplications and summations).
- f(x) = max(0, x): Changes all the negative values to 0. (Keeps the math from breaking.)
- ReLU layers train a lot faster than sigmoid without a significant reduction in accuracy.

CONVOLUTION: RELU (RECTIFIED LINEAR UNITS) LAYER




Deadpooling
POOLING

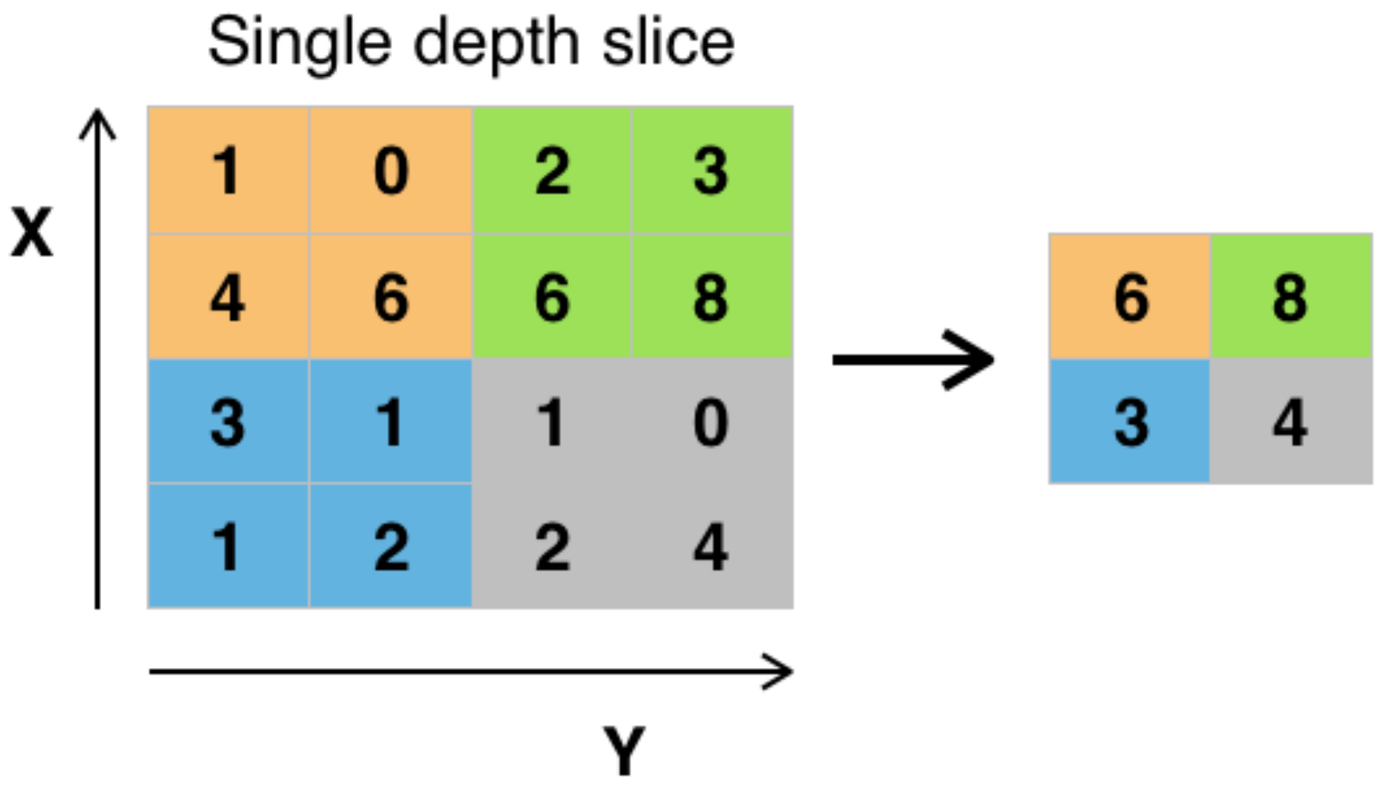
POOLING LAYERS

- Simplifies the information from the convolutional layer
- Images becomes smaller images
- One common procedure for pooling is known as *max-pooling*. In max-pooling, a pooling unit simply outputs the maximum activation in the 2×2 input region

POOLING LAYERS

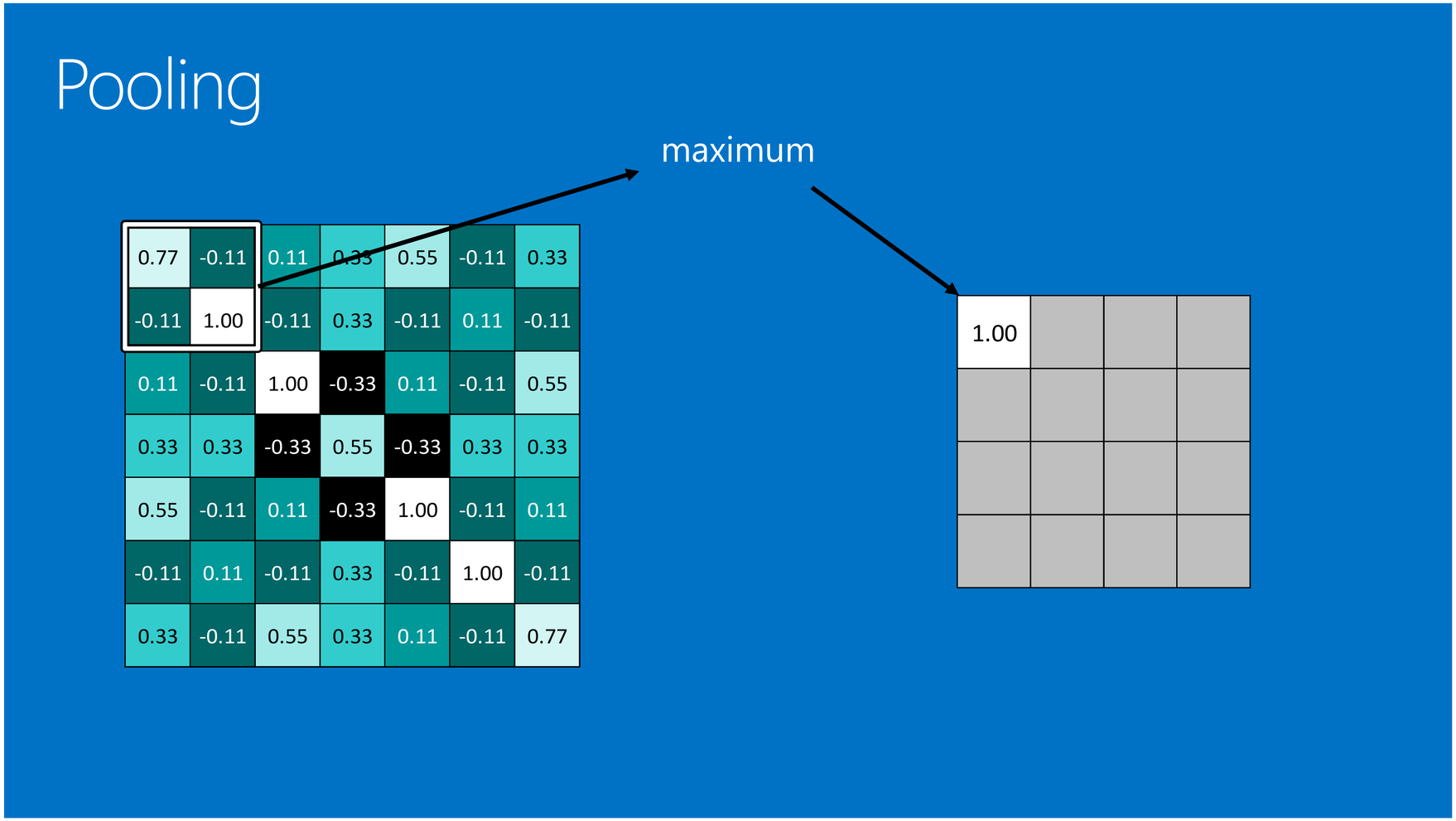
- We can think of max-pooling as a way for the network to ask whether a given feature is found anywhere in a region of the image.
- It then throws away the exact positional information.
- The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features.
- A big benefit is that there are many fewer pooled features, and so this helps reduce the number of parameters needed in later layers.

POOLING LAYERS

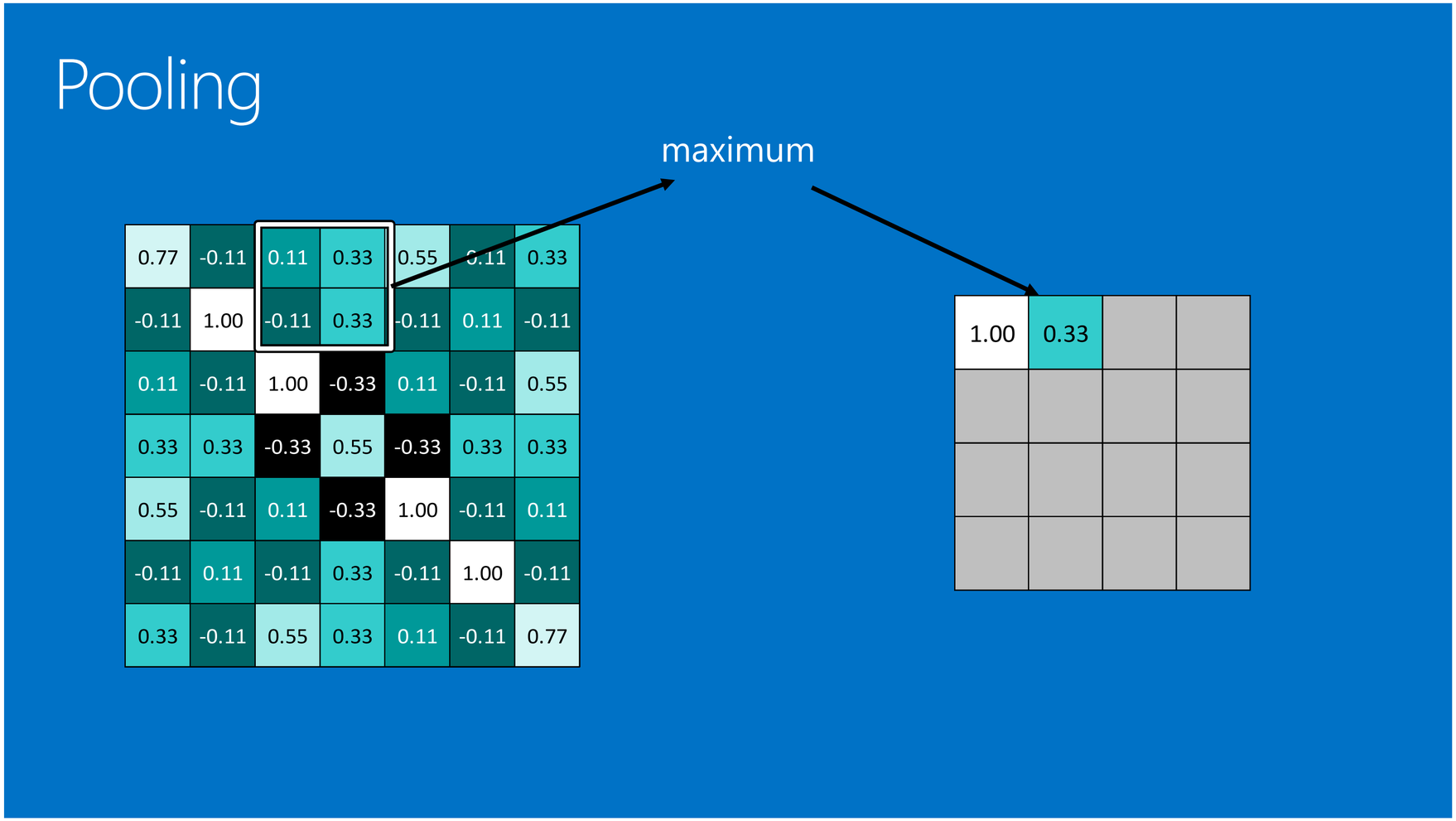


Example of Maxpool with a 2x2 filter and a stride of 2

POOLING LAYERS



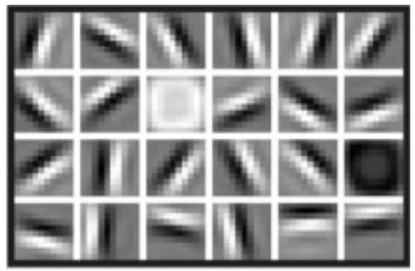
POOLING LAYERS



THE NETWORK

WHAT'S GOING ON IN THE LAYERS?

- Layer 1: Just pixels
- Layer 2: Identify edges and simple shapes
- Layer 3: More complex shapes and objects
- Layer 4: Learns which shapes can be used to define faces

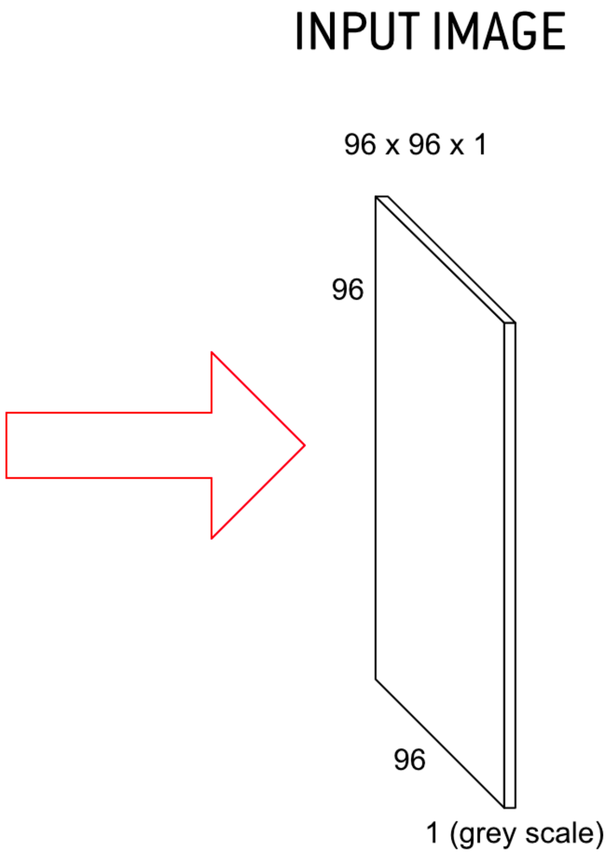
SO, THAT IS A CONVOLUTIONAL NEURAL NETWORK FOR PREDICTING FACIAL KEYPOINTS.

IMPLEMENTATION

IMPLEMENTATION

- “Steal like an artist”
- My solution is a mashup of what others have done

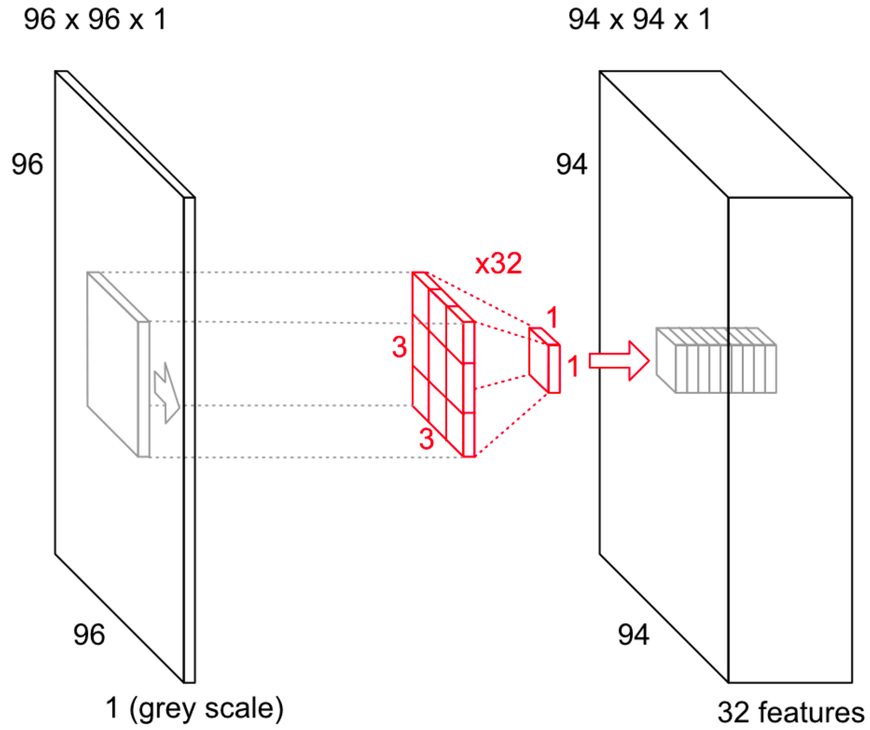
IMPLEMENTATION



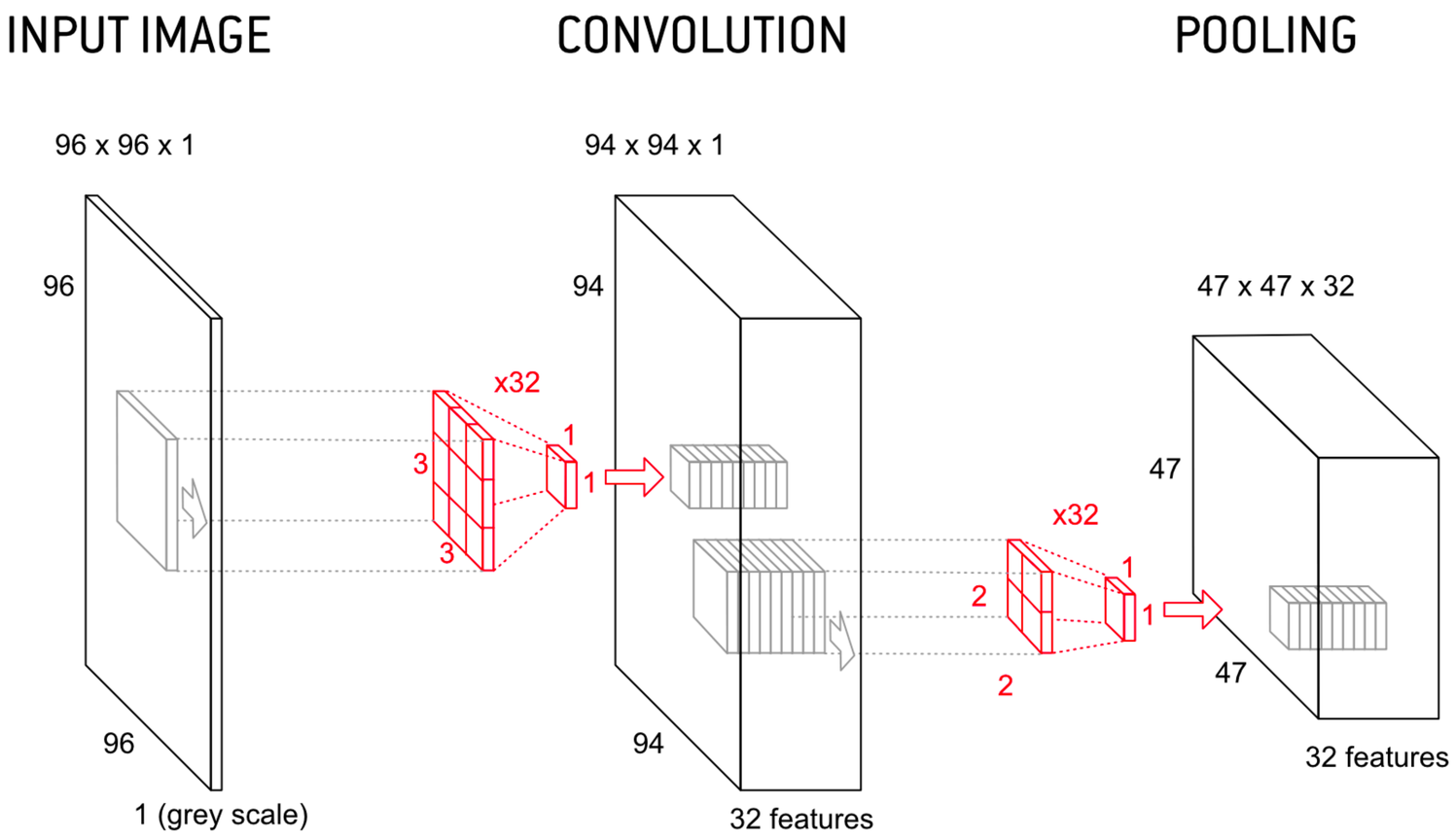
IMPLEMENTATION

INPUT IMAGE

CONVOLUTION

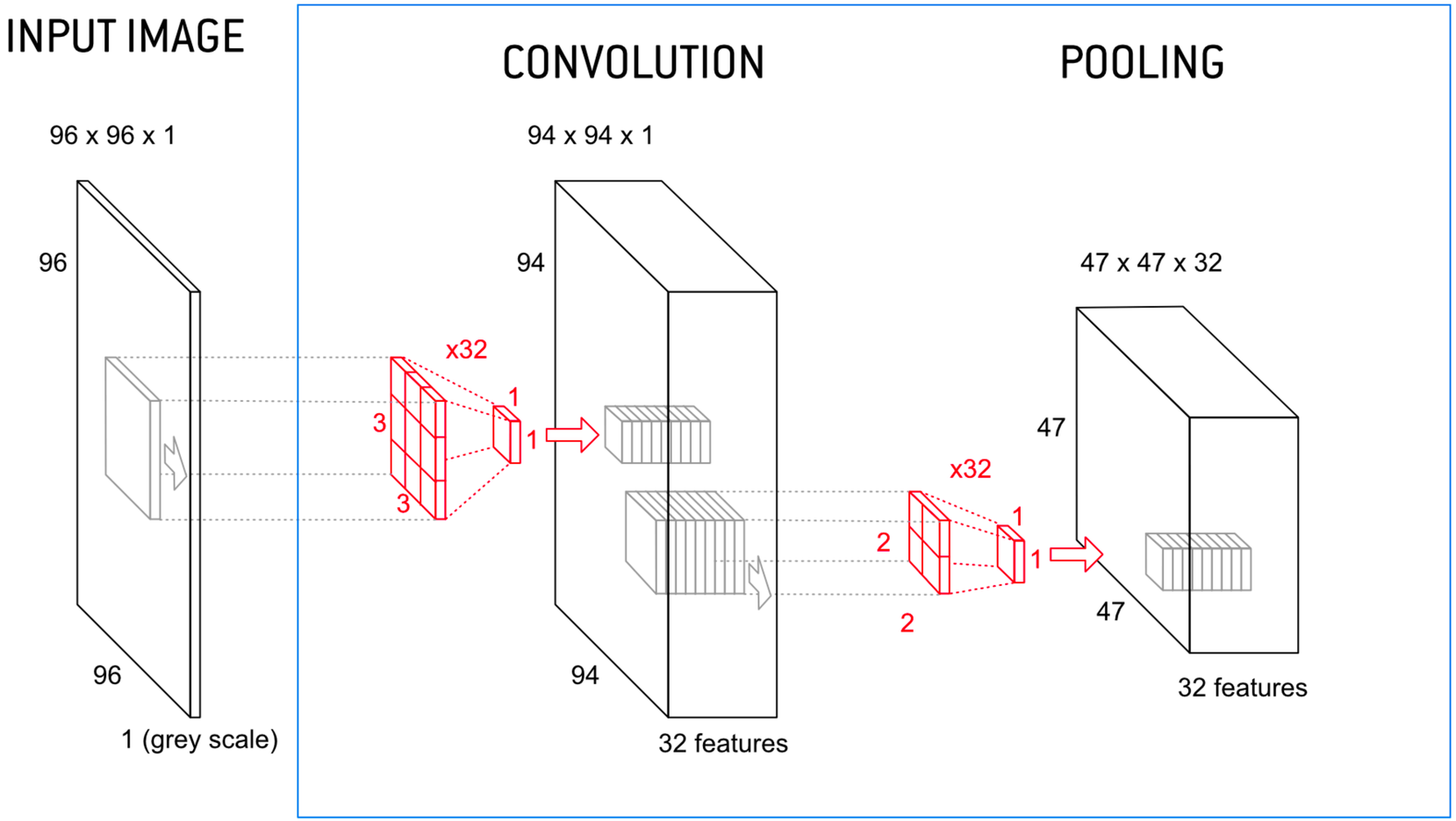


IMPLEMENTATION

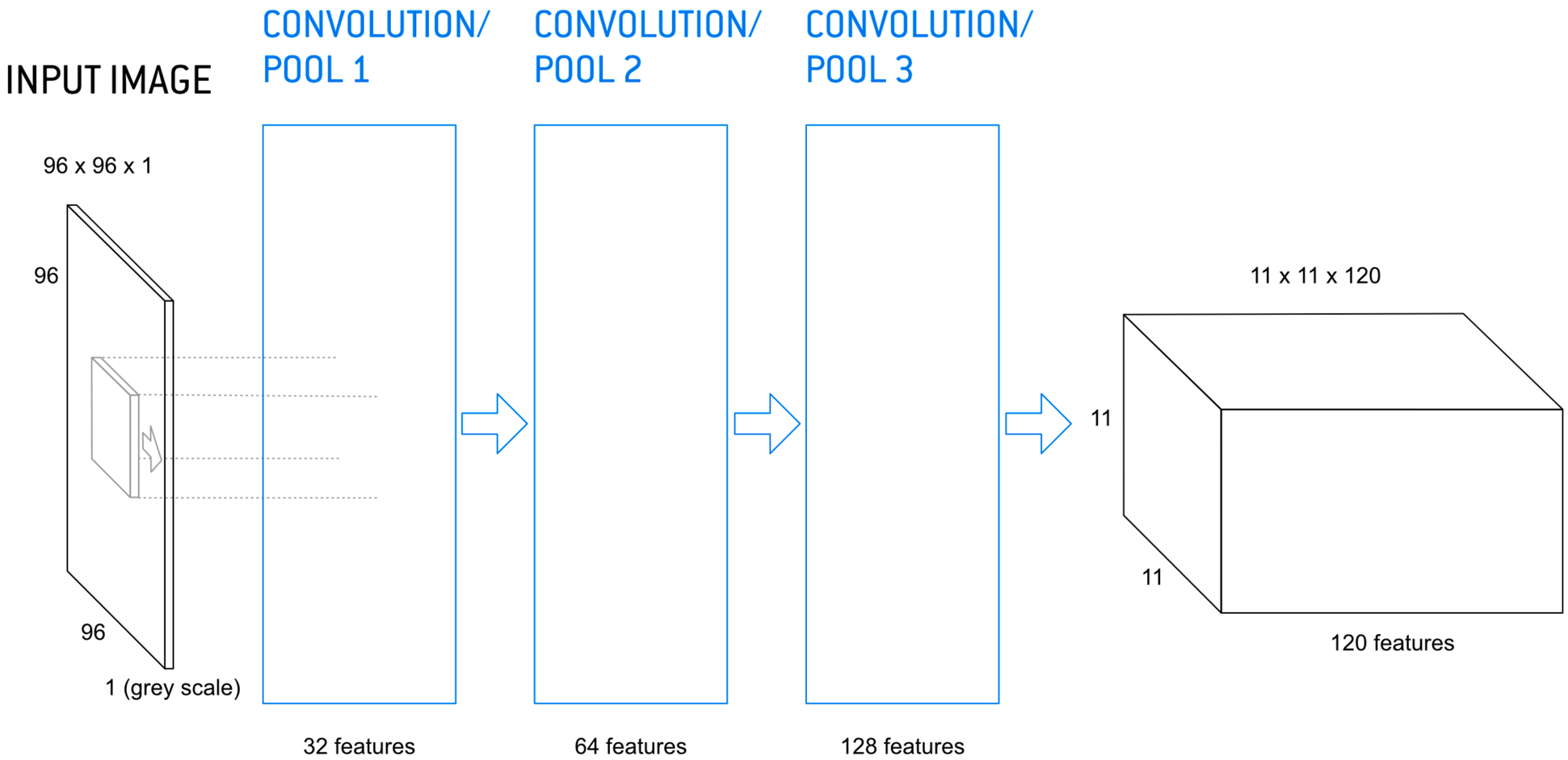


IMPLEMENTATION

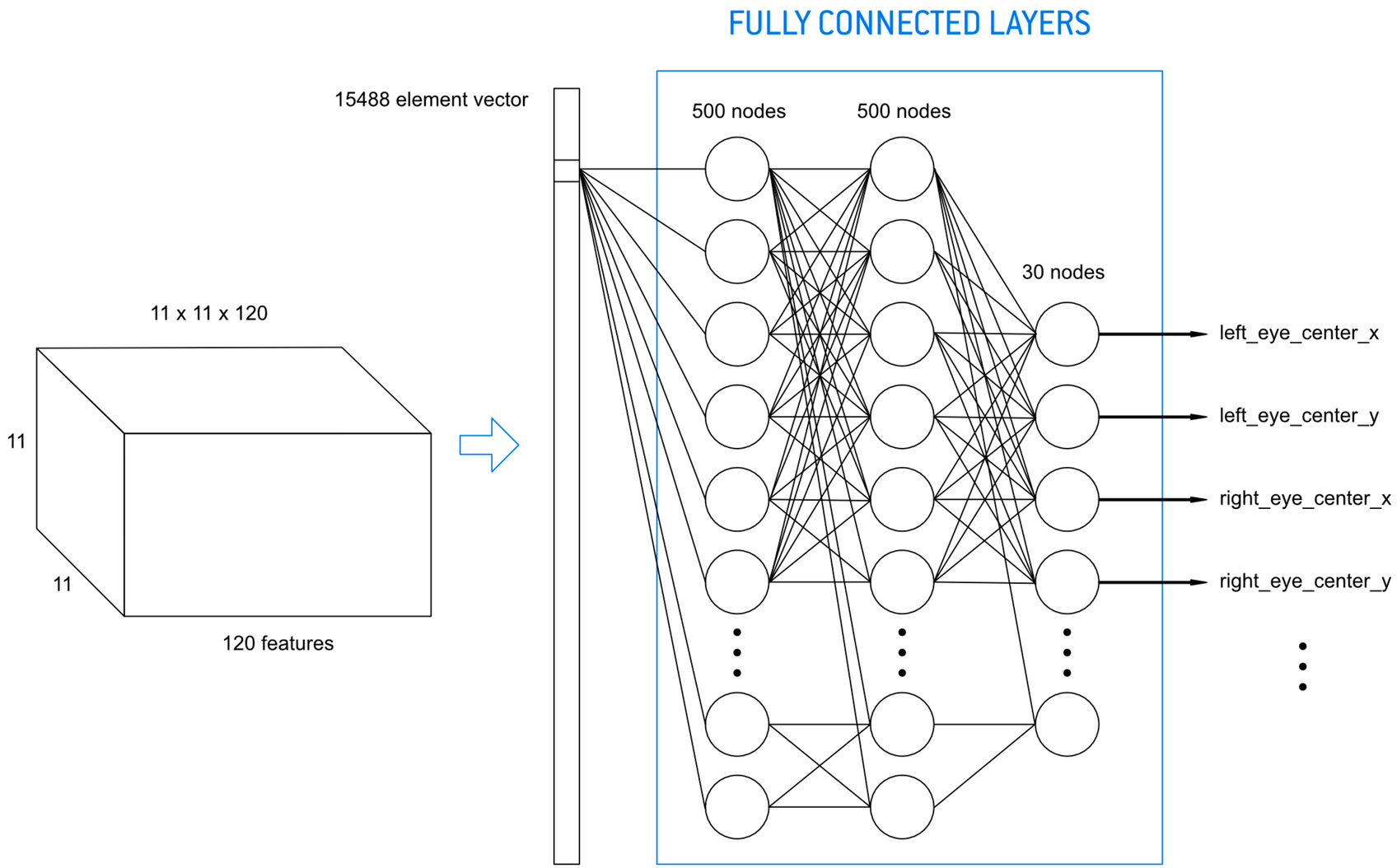
CONVOLUTION/POOL LAYER COMBO



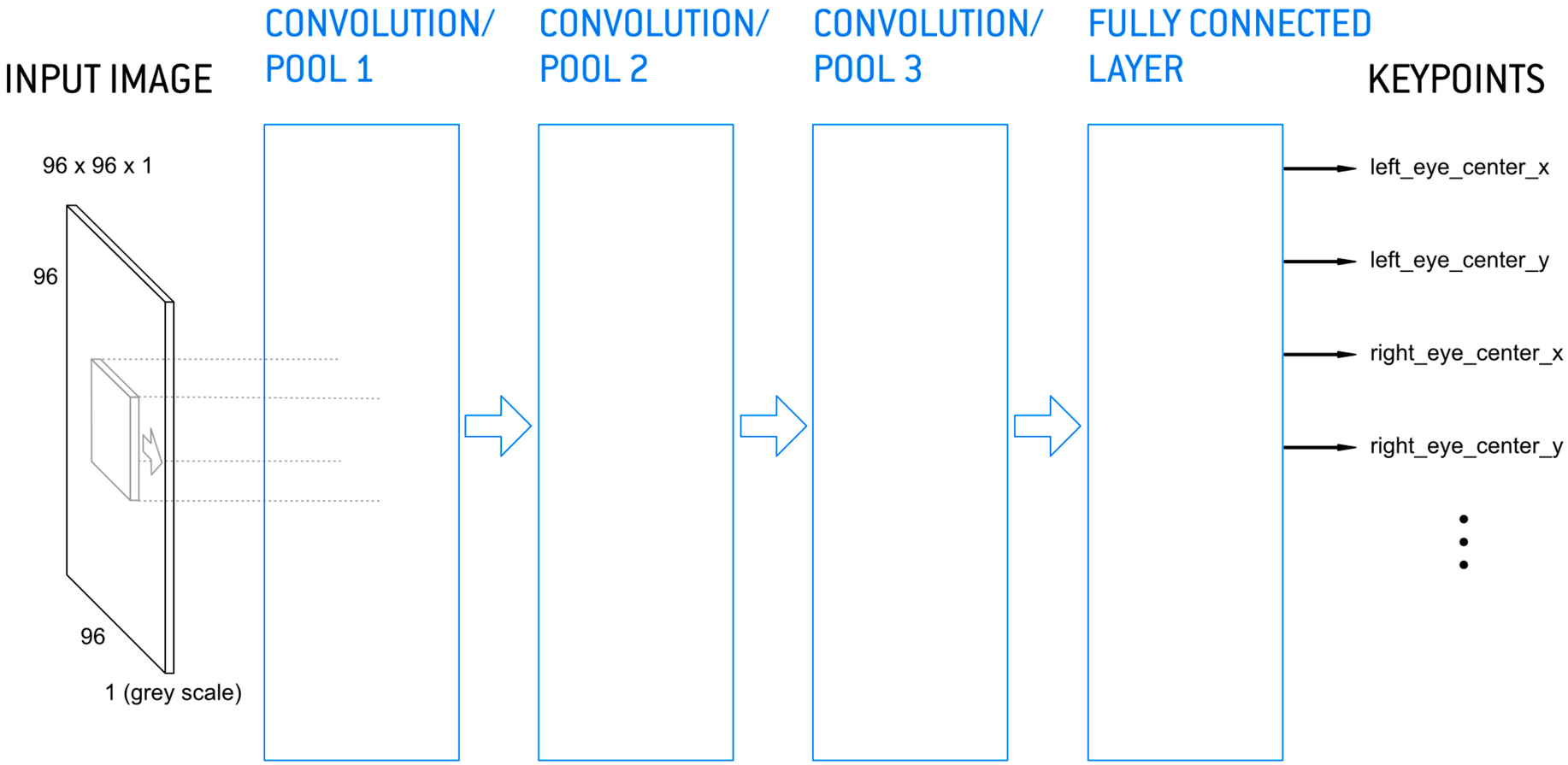
IMPLEMENTATION



IMPLEMENTATION



IMPLEMENTATION



IMPLEMENTATION

- TensorFlow & Scikit Flow
- “Steal like an artist”
- More work to do

MORE WORK TO DO

LEARNINGS

LEARNINGS

- CNNs are very capable and interesting solutions
- Many hyper-parameters
- Learn from others:
  - Kaggle forum members
  - Tutorials
  - Blogs

LEARNINGS

- You don’t have to be a superhero to use neural networks
- Be careful with who you get to help you with your data science projects

NEXT STEPS

NEXT STEPS

- Complete the project
- Submit entry to Kaggle
- More research
- More development (tune hyper-parameters)
- Hardware: Amazon EC2 (AWS)
- More Kaggle competitions

THANK YOU!

- GitHub repository with slides, paper and code: https://github.com//Khenu/sfdat26_work


LATER, NERDS!