Neural networks

Geir Storvik and Aliaksandr Hubin

April 2, 2022

Neural networks

- Flexible parametric models
- Universal approximators
- Successful in applications
- Image analysis
- Natural language processing
- No black magic same math as for Gaussian and/or Logistic regression
- But over-parametrization
- Lack of interpretability
- Difficult to reproduce the results due to non-concavity of the loss

Neural networks

- Principal components:

$$z_{ij} = \sum_{h=1}^{p} \alpha_{hj} x_{ih} = \boldsymbol{\alpha}_j^T \mathbf{x}_i \qquad j = 1, ..., J$$

$$y_i = \beta_0 + \sum_{j=1}^{J} \beta_j z_{ij} + \varepsilon_i = \underbrace{\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i}_{T_i} + \varepsilon_i$$

- $z_{ij}, j = 1, , ..., J$ can be considered as some latent (or hidden) variables.
- Generalization, One-layer Neural network:

$$z_{ij} = f_0(\boldsymbol{\alpha}_j^T \mathbf{x}_i), \qquad j = 1, ..., J \tag{1}$$

$$T_i = \beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i \tag{2}$$

$$y_i = f_1(T_i) + \varepsilon_i = f_1(\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i) + \varepsilon_i \tag{3}$$

where both $f_0(\cdot)$ and $f_1(\cdot)$ can be non-linear functions (aka **activation functions**).

Graphical display of 1-layer NN

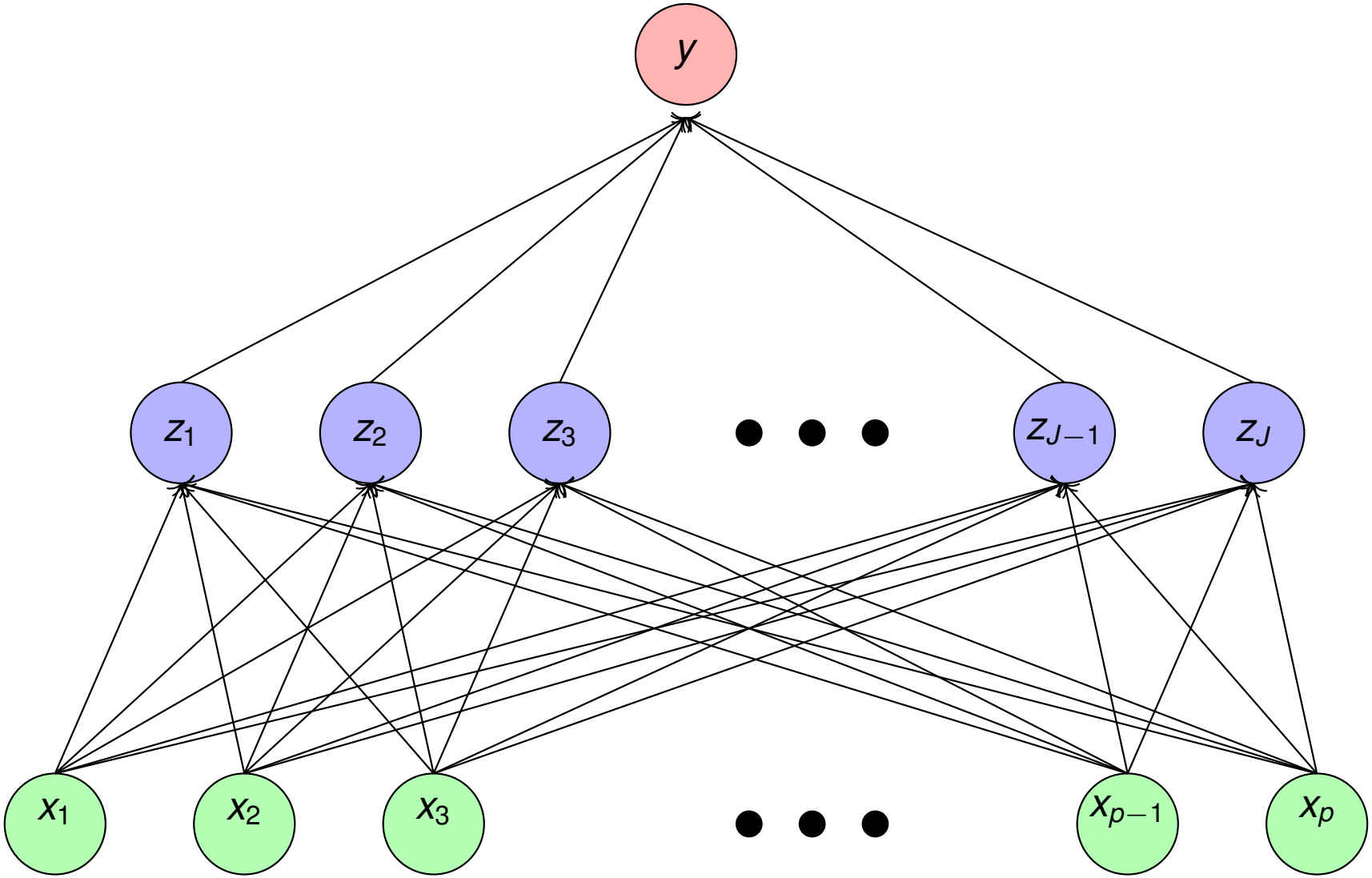
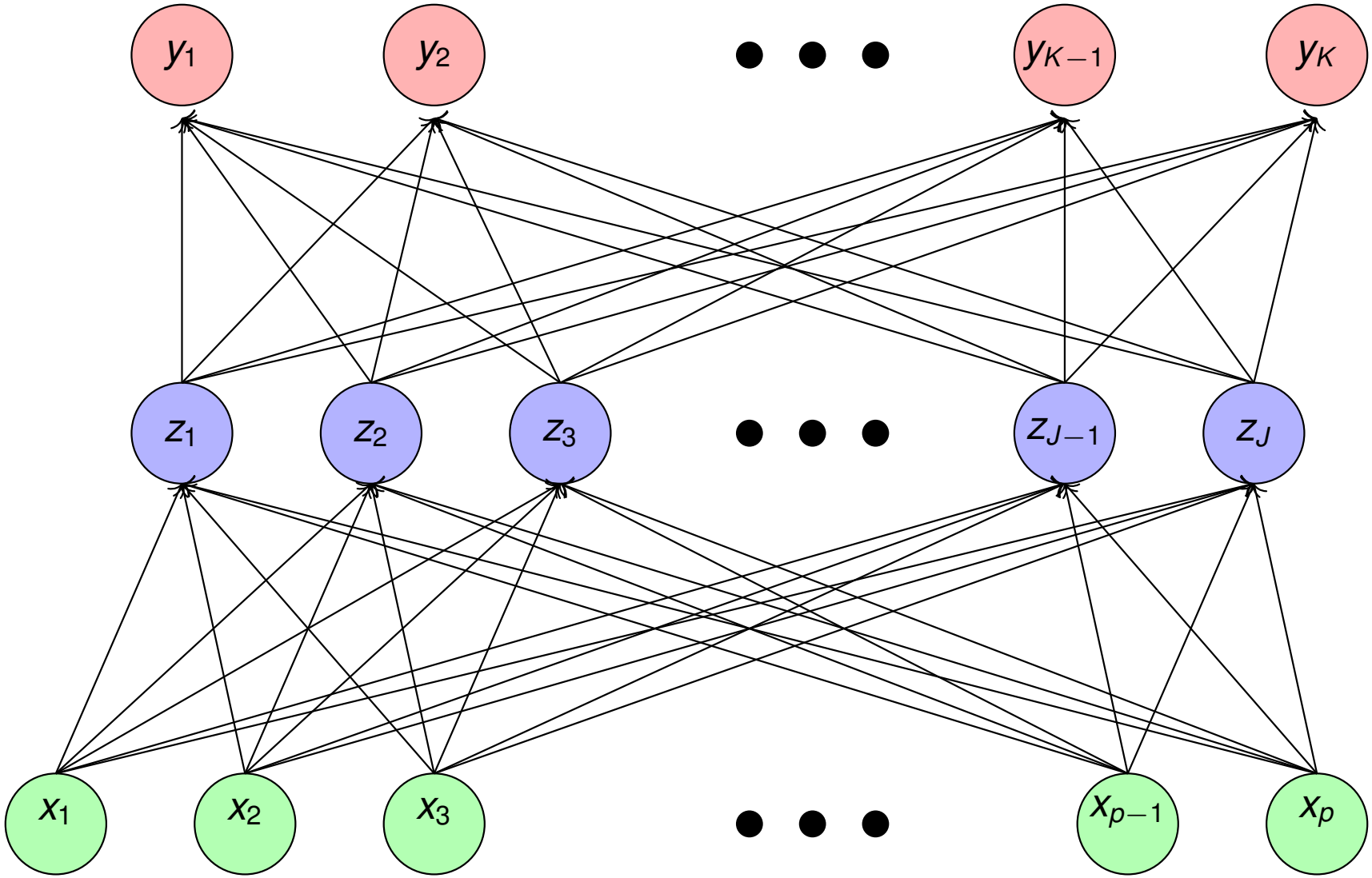


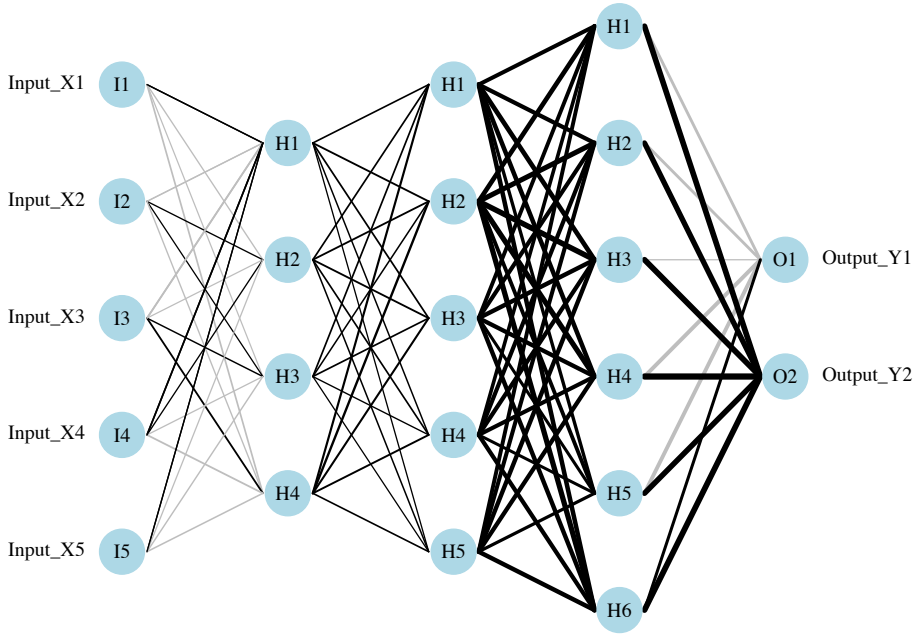
Figure: Visualisation of neural network with one hidden layer.

Neural net with K classes



Deep learning

- So far: Only one layer of latent variables
- Also possible with several layers



- Gives more flexibility, more possibilities for interactions and non-linearities
- Can give many parameters/weights
  - Example gives $5 * 4 + 4 * 5 + 5 * 6 + 6 * 2 = 82$ weights
  - With biases we have $5 * 4 + 4 + 4 * 5 + 5 + 5 * 6 + 5 + 6 * 2 + 2 = 99$ parameters
  - One layer with 15 latent variables gives $5 * 15 + 15 * 2 = 105$ weights or 120 parameters
- Overfitting also a problem here.

Properties neural networks

- Advantages
  - Flexibility
  - Impressive results in many large scale applications (image analysis, text processing)
  - Sequential upgrading
- Disadvantages
  - Can be time-consuming
  - Instability in estimation stage
  - Difficulties in choosing the number of latent variables, penalty value λ
  - Difficulties in obtaining standard errors and uncertainty measures
  - Interpretation
- Thus week
  - A powerful and flexible R library `torch`
  - Simple architectures
  - Maybe bit on Bayesian NNs if we have time

Common Layer Types: Linear Layers

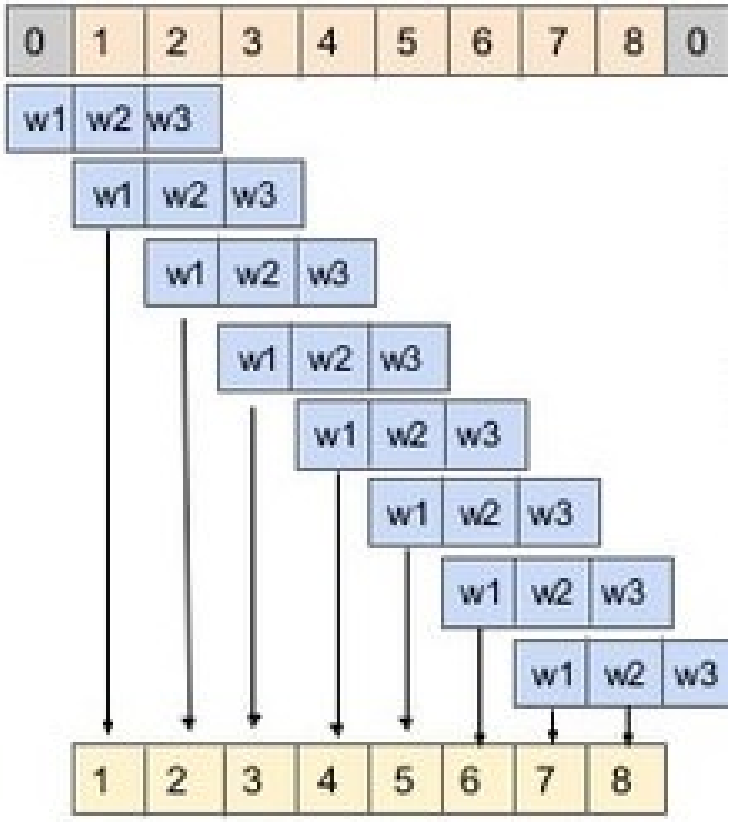
- The most basic type of neural network layer is a **linear** or **fully connected layer**.
- This is a layer where every input influences every output of the layer to a degree specified by the layer's weights.
- If a model has m inputs and n outputs, the weights will be an $m * x * n$ matrix.
- If you do the matrix multiplication of x by the linear layer's weights, and add the biases, you'll find that you get the output vector y.
- Linear layers are used widely in deep learning models.
- One of the most common places you'll see them is in classifier models, which will usually have one or more linear layers at the end, where the last layer will have n outputs, where n is the number of classes the classifier addresses.

Common Layer Types: Convolutional Layers

- Convolutional layers are built to handle data with a **high degree of spatial correlation.**
- They are very **commonly used in computer vision**, where they detect close groupings of features which the compose into higher-level features.
- The **first argument** to a convolutional layer's constructor is the number of input channels.
- A **convolutional layer** is a window that scans over the image and takes products of weights and values. These products are called features.
- The **second argument** to the constructor is the number of output features. In our example, we're asking our layer to learn 6 features.
- We likened the convolutional layer to a window - but how big is the window? The **third argument** is the window **or kernel size**.

Common Layer Types: Convolutional Layers (1d convolution)

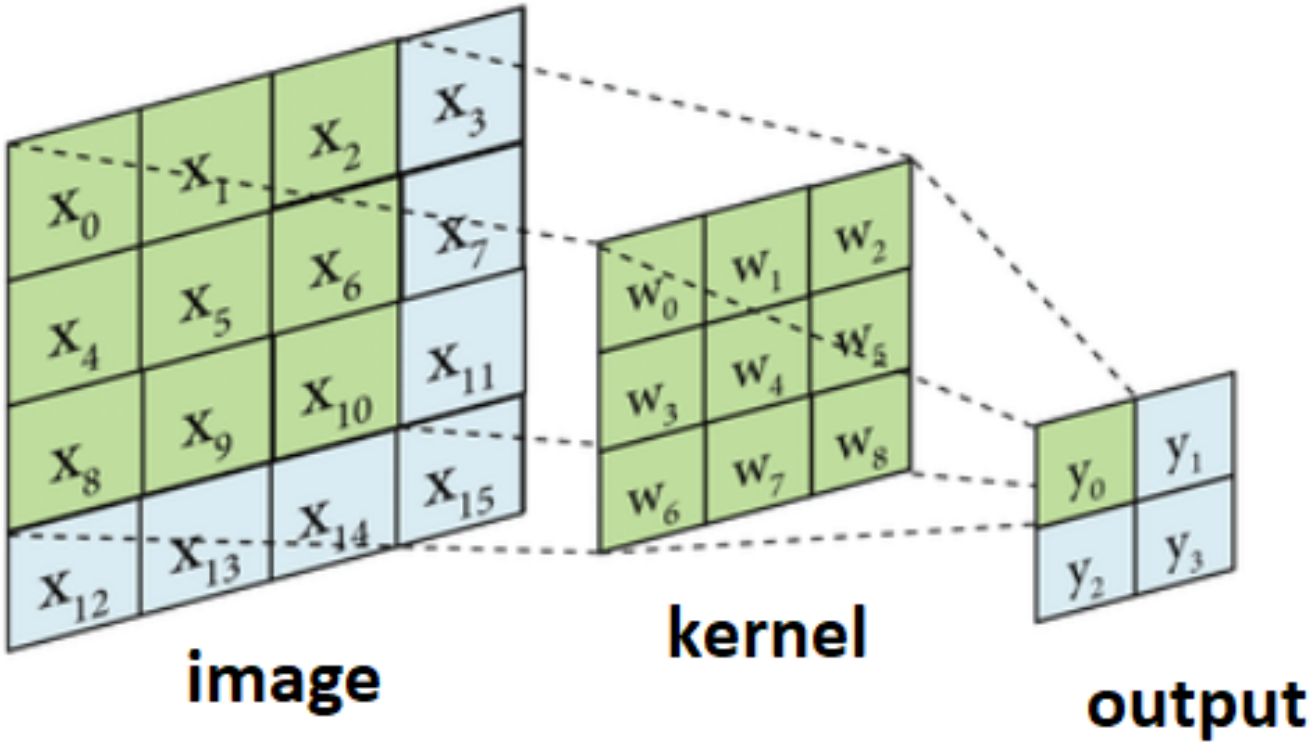
- Below **conv1(x,1,1,3,pad = 0, stride=1)** will give us an output tensor of 1x8; 1 is the number of features, and 8 is the number of features.
- **8** comes from the fact that when scanning a **3-pixel window** over a **8-pixel row**, there are only 8 valid positions.



Source: `https://ai-pool.com/d/1d-convolution-in-keras`

Common Layer Types: Convolutional Layers (2d convolution)

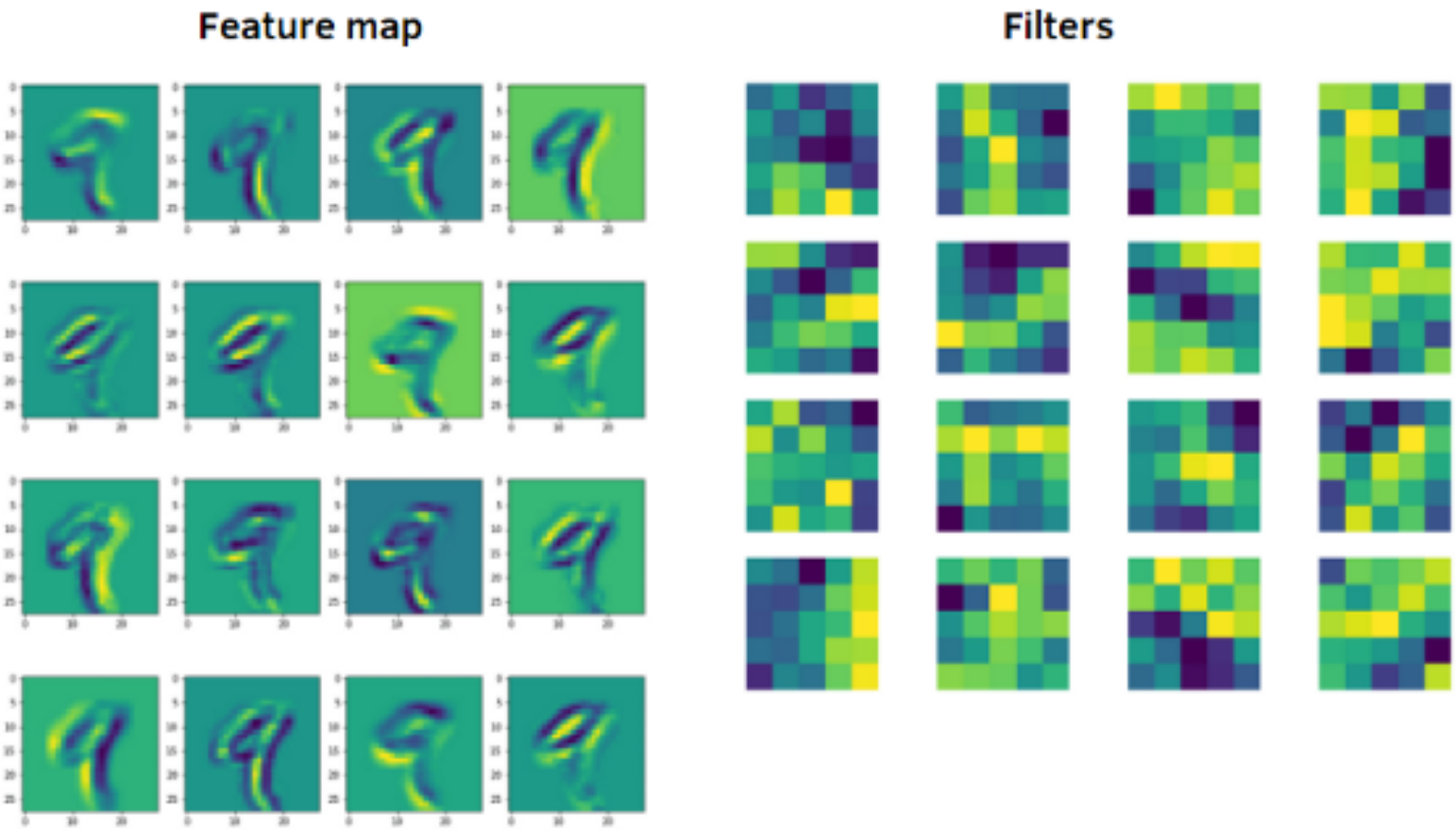
- **conv2(x,inch=1,outch = 1,c(3,3)),pad = none, stride=1)** will give us an output tensor of 1x8; 1 is the number of features, and 8 is the number of features.
- **8** comes from the fact that when scanning a **3-pixel window** over a **8-pixel row**, there are only 8 valid positions.



Source: `https://medium.com/@thepyprogrammer/2d-image-convolution-with-numpy-with-a-handmade-sliding-window-view-94`

Common Layer Types: Convolutional Layers (2d convolution)

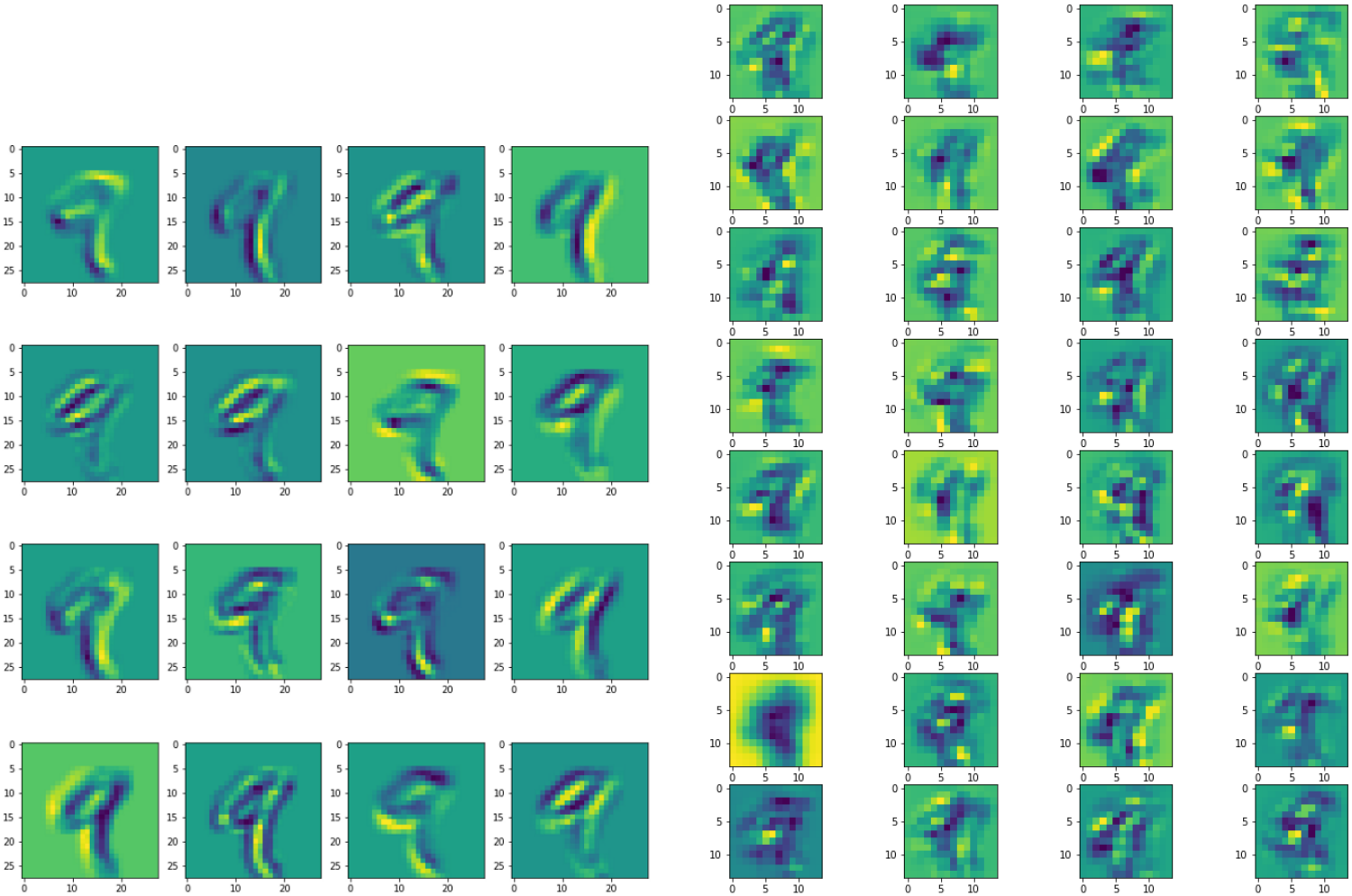
- Illustration:



Source: `https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-netwo`

Common Layer Types: Convolutional Layers (2d convolution)

- Illustration:

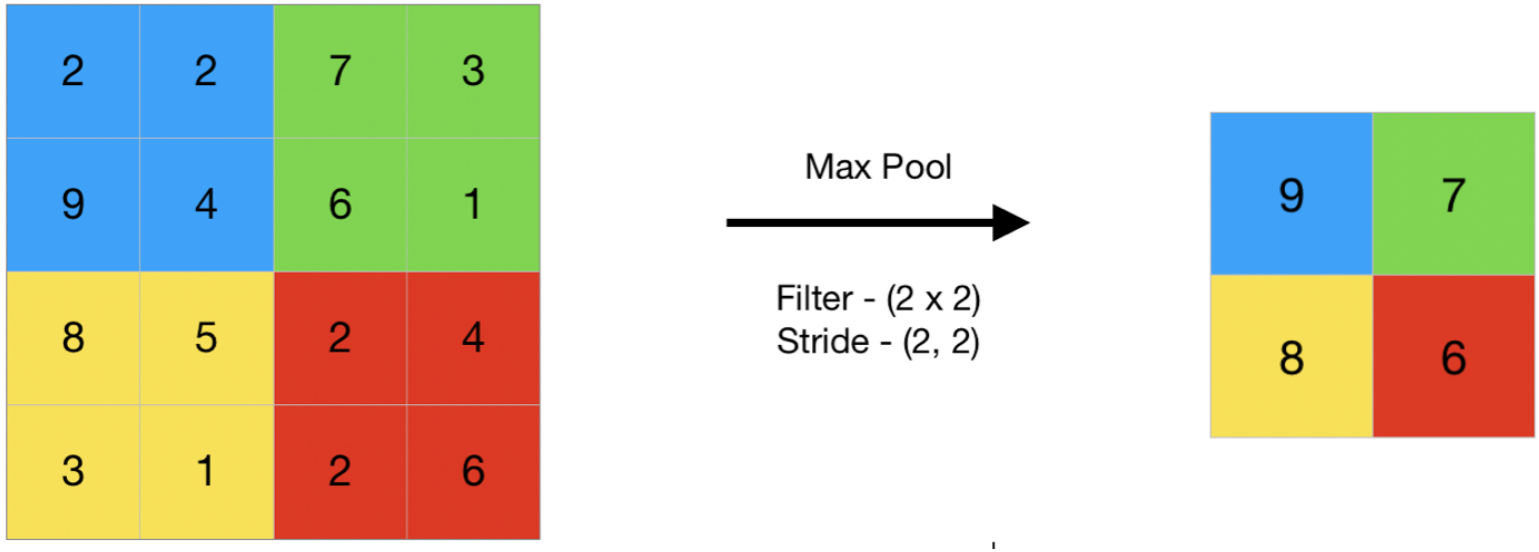


Source:

https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-netwo

Common Layer Types: Max pooling

- Max pooling (and its twin, min pooling) reduce a tensor by combining cells, and assigning the maximum value of the input cells to the output cell.

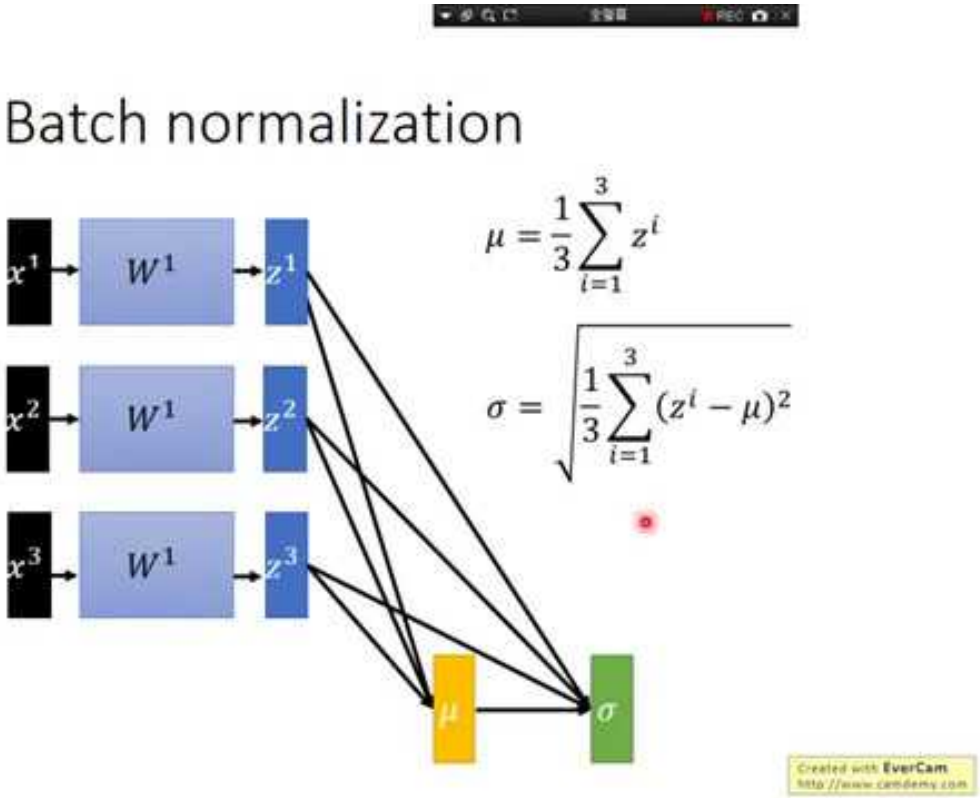


Source:

`https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/`

Common Layer Types: Normalization layers

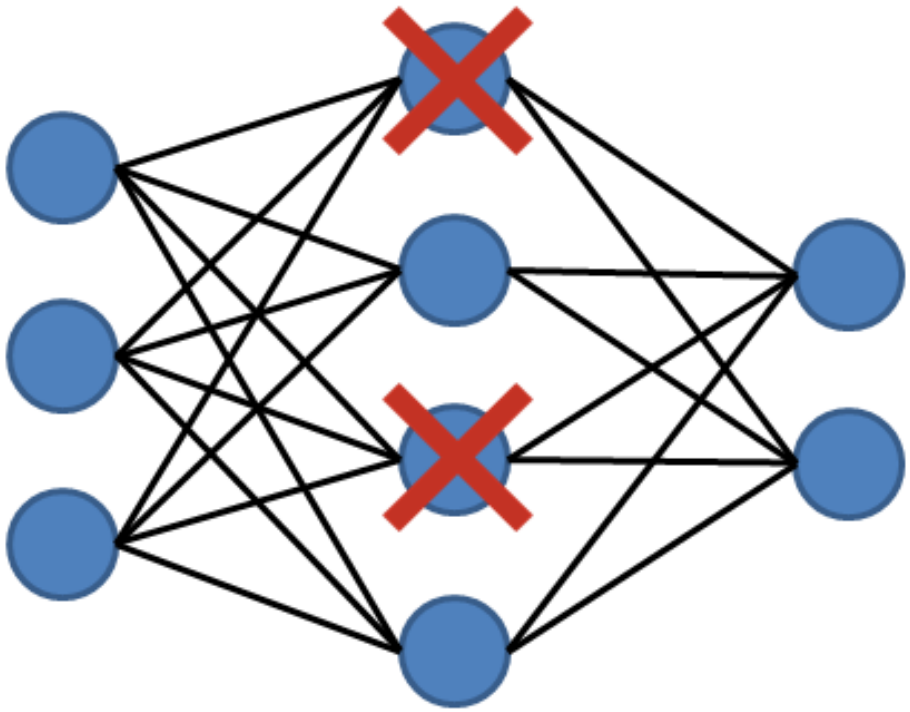
- Normalization layers re-center and normalize the output of one layer before feeding it to another.
- Centering the and scaling the intermediate tensors has a number of beneficial effects, such as fair regularization discussed in the previous lecture.



Source: `https://www.youtube.com/watch?v=BZh1ltr5Rkg`

Common Layer Types: Dropout layers

- Dropout layers are a tool for turning GD/SGD to coordinate-wise GD/SGD.
- Dropout layers work by randomly turning to zero parts of the gradients.



Source: `https://towardsdatascience.com/coding-neural-network-dropout-3095632d25ce`

Torch main feature: Aurograd

- Neural networks (NNs) are a collection of nested functions that are executed on some input data.
- These functions are defined by parameters (consisting of weights and biases), which in PyTorch are stored in tensors.
- Training a NN happens in two steps:
- **Forward Propagation:** In forward prop, the NN makes its best guess about the correct output. It runs the input data through each of its functions to make this guess.
- **Backward Propagation:** In backprop, the NN adjusts its parameters proportionate to the error in its guess. It does this by **traversing backwards from the output, collecting the derivatives of the error with respect to the parameters** of the functions (gradients).

Data-loaders

- In deep learning models we don't usually have all your data in RAM
- But because we are usually training using mini-batch gradient descent thus only needing a mini-batch on RAM each time.
- In torch we use the '**datasets**' abstraction to define the process of loading data.
- Once you have defined your dataset you can use torch dataloaders that allows you to iterate over this dataset in batches.
- Note that datasets are optional in torch. They are jut there as a recommended way to load data.
- We shall see an example of how to create a simple torch dataset that pre-process a data.frame into tensors so we can feed them to a model.

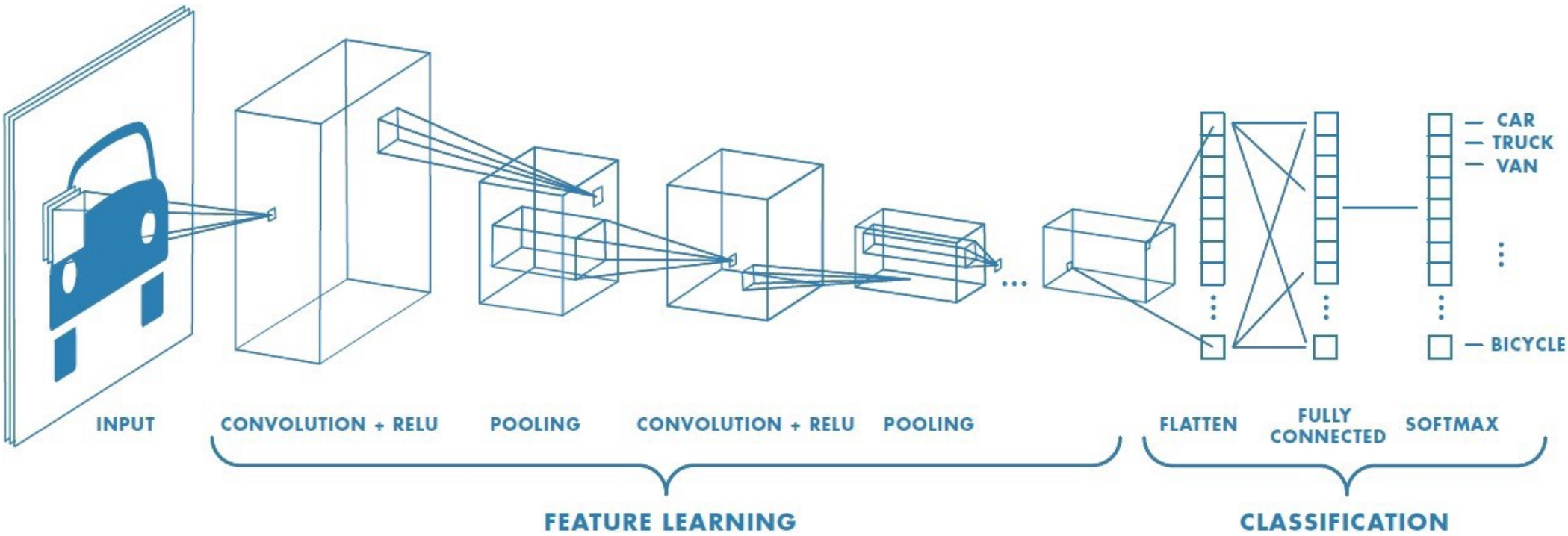
Examples

- A simple differentiable function $ax + b0$.
- Regression on 'penguins' data set
  - Manual optimization algorithm;
  - Out-of-the box optimizer.
- Classification on 'penguins' data set
  - Out-of-the box optimizer.
- Classification of MNIST data
  - Dense deep neural network;
  - Convolutional neural network.

CNN

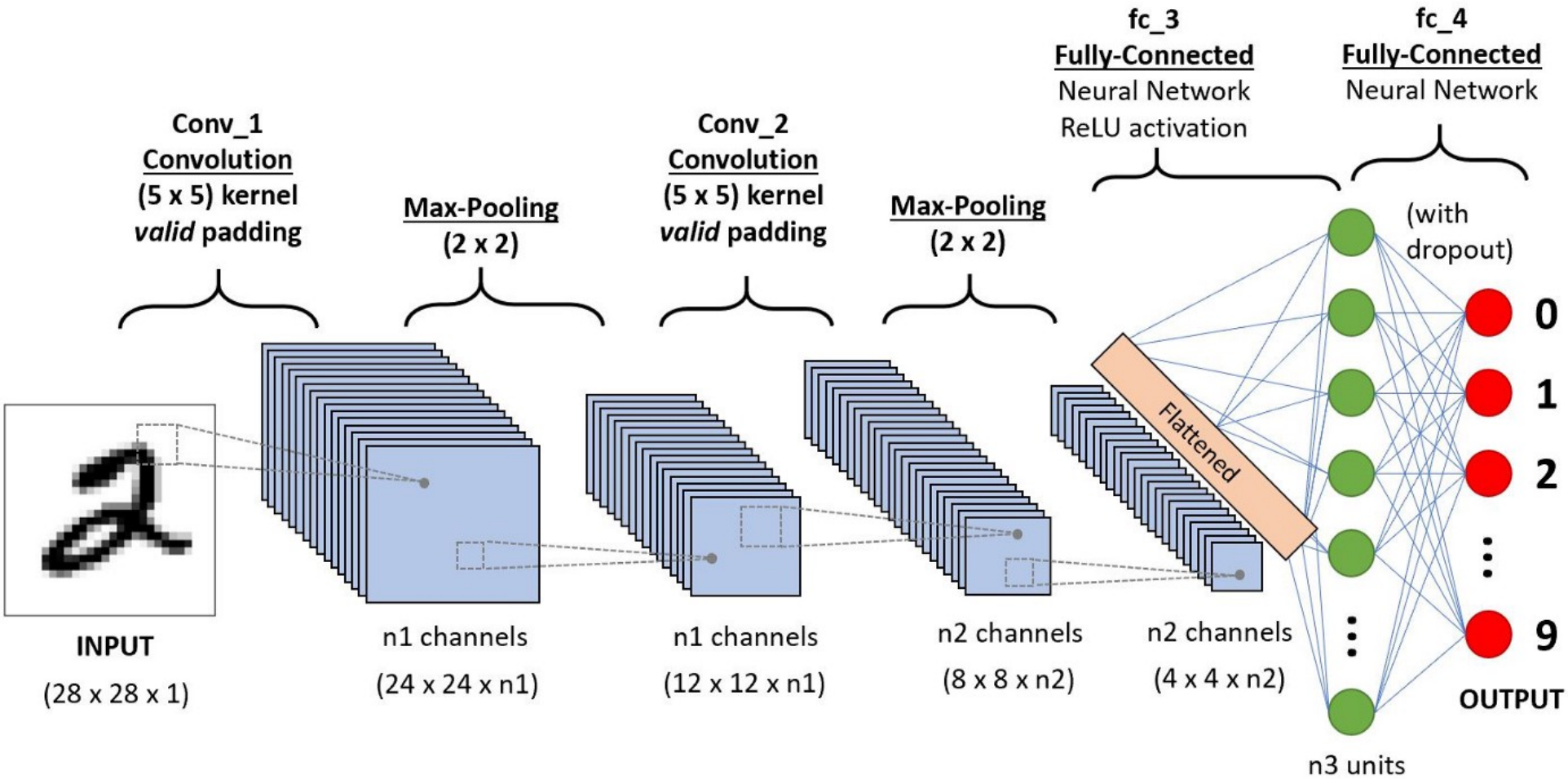
- A convolutional network is able to capture the spatial dependencies in an image through filters.
- It performs a better fitting to the image dataset due to the reduction in the number of parameters involved and re-usability of weights.
- The first ConvLayer captures the Low-Level features such as edges, color, gradient orientation, etc.
- With added layers, the architecture adapts to the High-Level features as well
- Giving us a network which has a better representation of images in the dataset.

CNN



Source: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3b`

CNN for hand written digits recognition



Source: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3b`