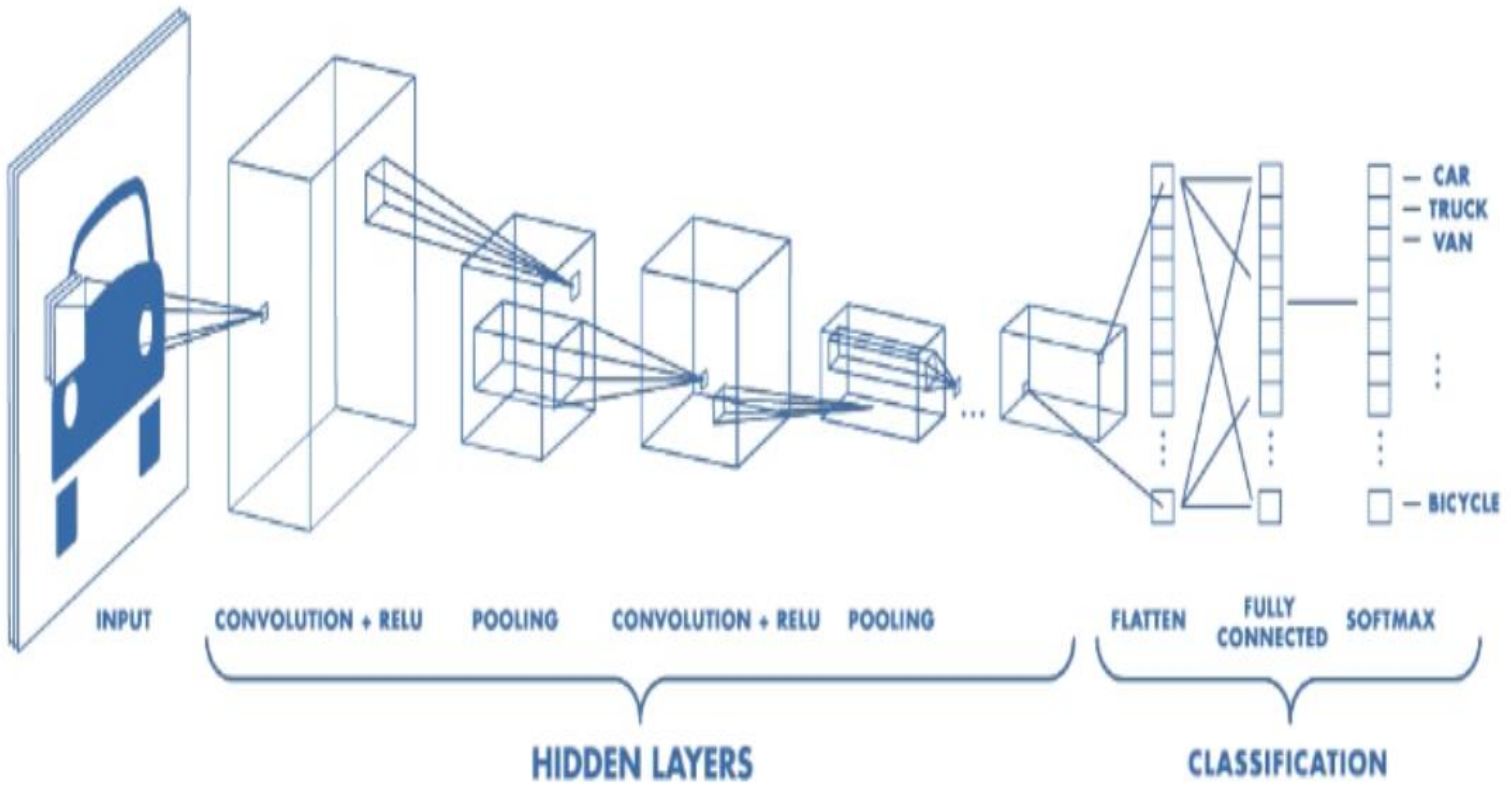
CNN &TRANSFER LEARNING

Lab6

CNN

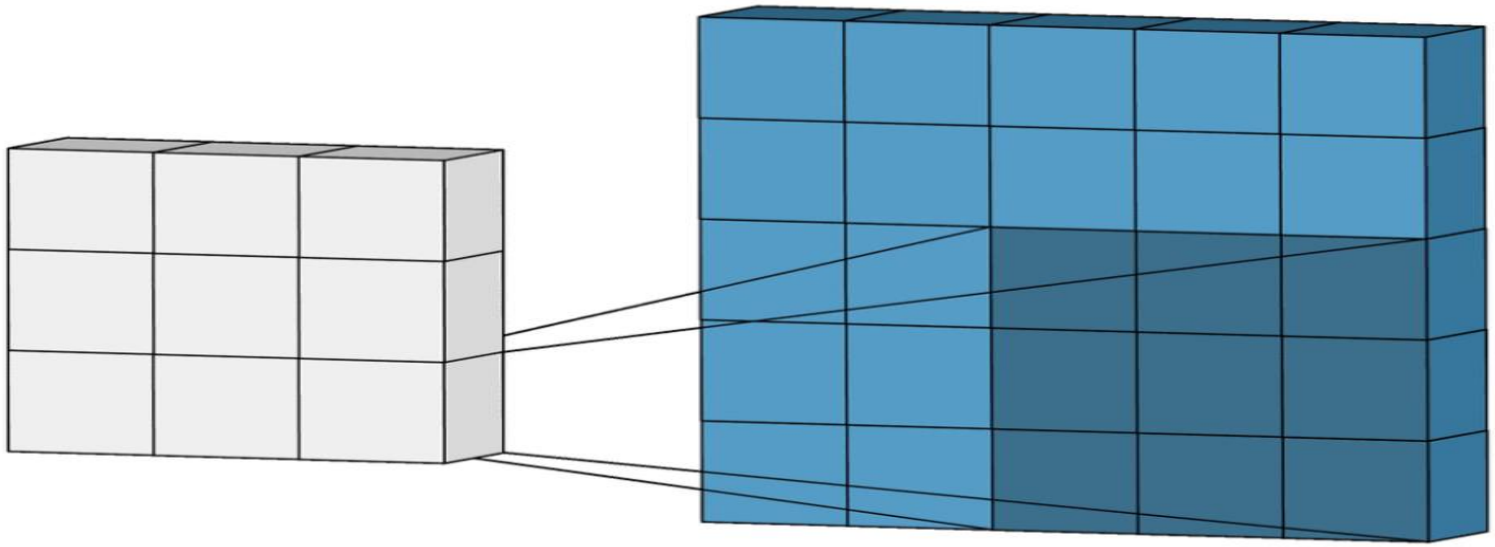
- A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing image.
- A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

CNN ARCHITECTURE



Convolution Layer

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth.



CONVOLUTION LAYER

If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

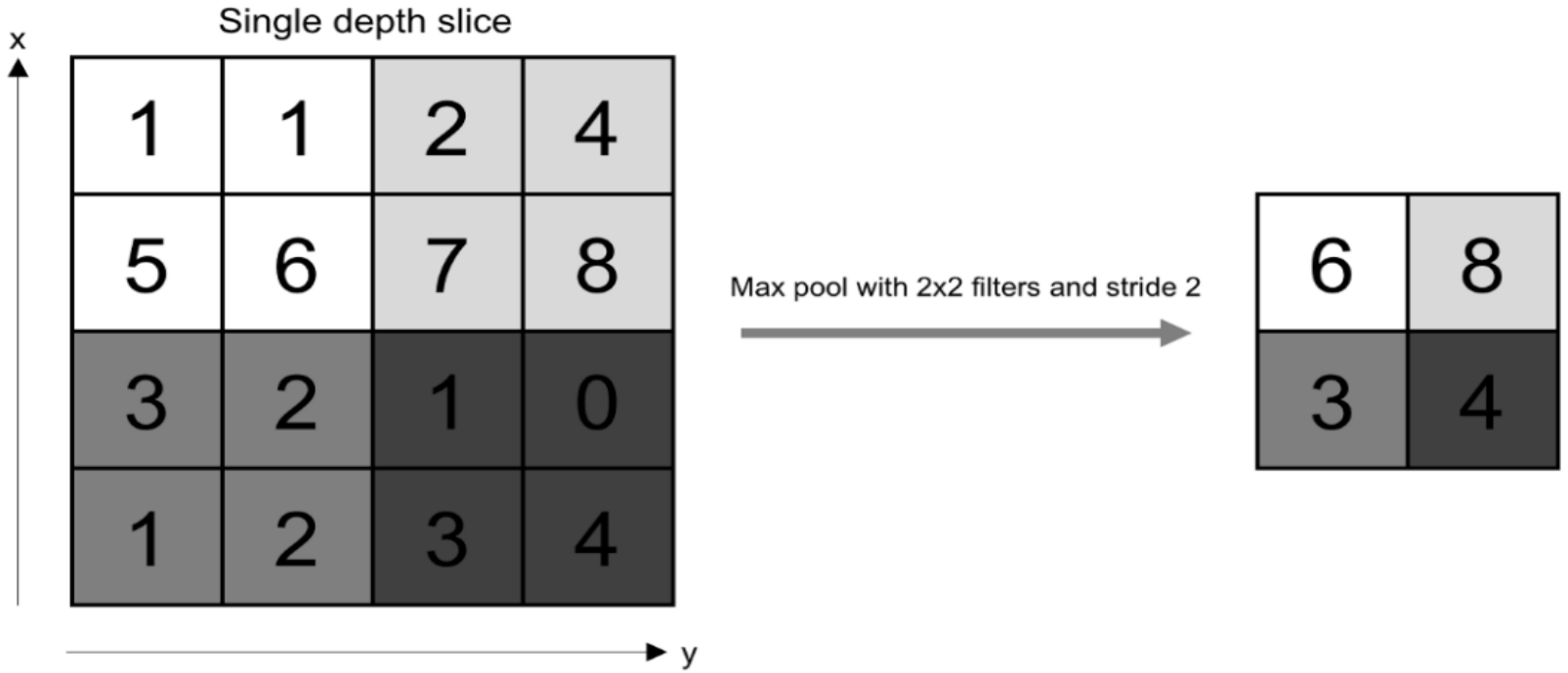
This will yield an output volume of size *Wout* x *Wou*t x *Dout*.

MOTIVATION BEHIND CONVOLUTION

- Convolution leverages three important ideas that motivated computer vision researchers:
- **sparse interaction:** This is achieved by making kernel smaller than the input
- **parameter sharing:** In a traditional neural network, each element of the weight matrix is used once and then never revisited, while convolution network has *shared parameters* i.e., for getting output, weights applied to one input are the same as the weight applied elsewhere.
- **equivariant representation:** It says that if we changed the input in a way, the output will also get changed in the same way.

POOLING LAYER

- The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights



POOLING

If we have an activation map of size *W* x *W* x *D*, a pooling kernel of spatial size *F*, and stride *S*, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F}{S} + 1$$

This will yield an output volume of size *Wout* x *Wout* x *D*.
In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

FULLY CONNECTED LAYER

- Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN.
- The FC layer helps to map the representation between the input and the output.

NON-LINEAR LAYER

- Non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map.
- There are several types of non-linear operations, the popular ones being:
- **Sigmoid**-The sigmoid non-linearity has the mathematical form $\sigma(\kappa) = 1/(1+e^{-}\kappa)$. It takes a real-valued number and “squashes” it into a range between 0 and 1.
- **Tanh**-Tanh squashes a real-valued number to the range [-1, 1]. Like sigmoid, the activation saturates, but — unlike the sigmoid neurons — its output is zero centered.
- **ReLU**- It computes the function $f(\kappa)=\max(0,\kappa)$. In other words, the activation is simply threshold at zero.

DESIGNING A CNN

- We will be using Fashion-MNIST, which is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

  Our convolutional neural network has architecture as follows:
  [INPUT]
  →[CONV 1] → [BATCH NORM] → [ReLU] → [POOL 1]
  → [CONV 2] → [BATCH NORM] → [ReLU] → [POOL 2]
  → [FC LAYER] → [RESULT]
  For both conv layers, we will use kernel of spatial size 5 x 5 with stride size 1 and padding of 2. For both pooling layers, we will use max pool operation with kernel size 2, stride 2, and zero padding.

DESIGNING A CNN

CONV 1
Input Size (W_1 x H_1 x D_1) = 28 x 28 x 1
• Requires four hyperparameter: ○ Number of kernels, k = 16 ○ Spatial extend of each one, F = 5 ○ Stride Size, S = 1 ○ Amount of zero padding, P = 2
• Outputting volume of W_2 x H_2 x D_2 ○ W_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ H_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ D_2 = k
Output of Conv 1 (W_2 x H_2 x D_2) = 28 x 28 x 16

DESIGNING A CNN

POOL 1

Input Size (W_2 x H_2 x D_2) = 28 x 28 x 16

- Requires two hyperparameter:
  - Spatial extend of each one, F = 2
  - Stride Size, S = 2

- Outputting volume of W_3 x H_3 x D_2
  - $W_3 = (28 - 2) / 2 + 1 = 14$
  - $H_3 = (28 - 2) / 2 + 1 = 14$

Output of Pool 1 (W_3 x H_3 x D_2) = 14 x 14 x 16

DESIGNING A CNN

CONV 2
Input Size (W_3 x H_3 x D_2) = 14 x 14 x 16
• Requires four hyperparameter: o Number of kernels, k = 32 o Spatial extend of each one, F = 5 o Stride Size, S = 1 o Amount of zero padding, P = 2
• Outputting volume of W_4 x H_4 x D_3 o W_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o H_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o D_3 = k
Output of Conv 2 (W_4 x H_4 x D_3) = 14 x 14 x 32

DESIGNING A CNN

POOL 2
Input Size (W_4 x H_4 x D_3) = 14 x 14 x 32
• Requires two hyperparameter: ○ Spatial extend of each one, F = 2 ○ Stride Size, S = 2
• Outputting volume of W_5 x H_5 x D_3 ○ W_5 = (14 – 2) / 2 + 1 = 7 ○ H_5 = (14 – 2) / 2 + 1 = 7
Output of Pool 2 (W_5 x H_5 x D_3) = 7 x 7 x 32

DESIGNING A CNN

FC Layer
Input Size (W_5 x H_5 x D_3) = 7 x 7 x 32
Output Size (Number of Classes) = 10

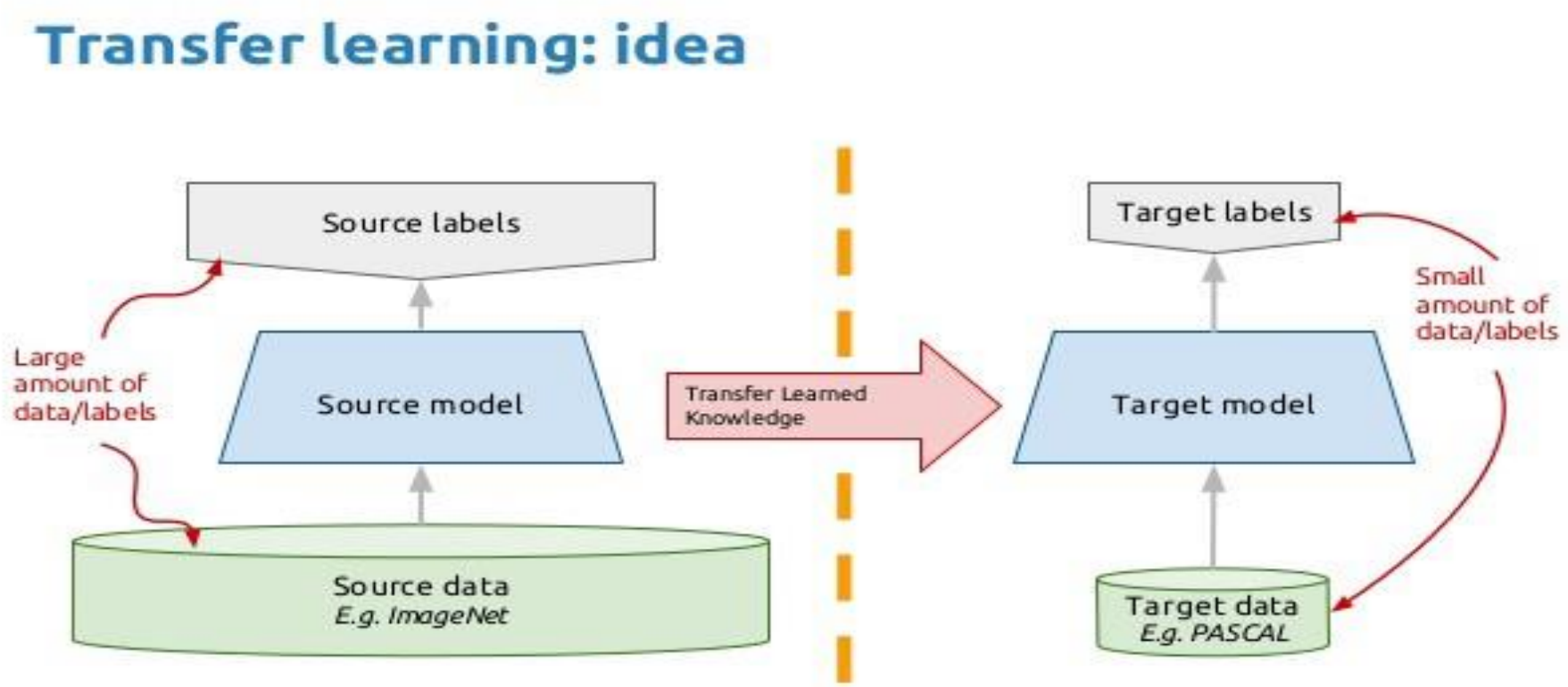
TRANSFER LEARNING

Training a Convolutional Neural Network from scratch poses many challenges, most notably the **amount of data** to train the network and the **amount of time** it takes for training to take place.

Transfer learning is a technique that allows us to use a model trained for a certain task as a starting point for a machine learning model for a different task.

For example, suppose a model is trained for image classification on the ImageNet dataset. In that case, we can take this model and “re-train” it to recognize classes it was *never* trained to recognize in the first place!

TRANSFER LEARNING



TRANSFER LEARNING METHODS

There are two primary types of transfer learning:
1.Transfer learning via feature extraction: We remove the FC layer head from the pre-trained network and replace it with a softmax classifier. Here we treat the pre-trained CNN as a feature extractor and then pass those features through a Logistic Regression classifier.

1.Transfer learning via fine-tuning: When applying fine-tuning, we again remove the FC layer head from the pre-trained network, but this time we construct a *brand new, freshly initialized FC layer head* and place it on top of the original body of the network. The weights in the body of the CNN are frozen, and then we train the new layer head (typically with a very small learning rate). We may then choose to unfreeze the body of the network and train the *entire* network.

The first method tends to be easier to work with, as there is less code involved and fewer parameters to tune.
However, the second method tends to be more accurate, leading to models that generalize better.

TRANSFER LEARNING STEPS FOR OBJECT RECOGNITION

Following is the general outline for transfer learning for object recognition:

1.Load in a pre-trained CNN model trained on a large dataset
2.Freeze parameters (weights) in model's lower convolutional layers
3.Add custom classifier with several layers of trainable parameters to model
4.Train classifier layers on training data available for task
5.Fine-tune hyperparameters and unfreeze more layers as needed