RWTH AACHEN UNIVERSITY

How to Teach "Convolutional Neural Networks"

Stephanie Käs

M. Sc. Physics

HybridLaunch Gießen n

Hybrid Rocket Developme

Speaker & Marketing Manager & Graphic

AI Center RWTH Aac

Computer Vision & Machine

Research & Teaching A

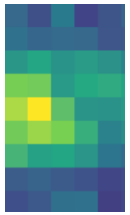
JLU Gie

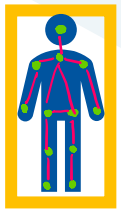
Statistik für Ge

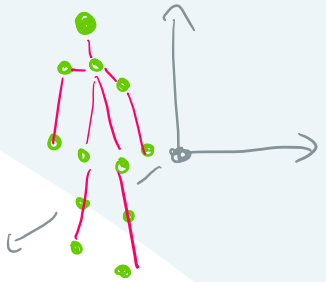
Current Research

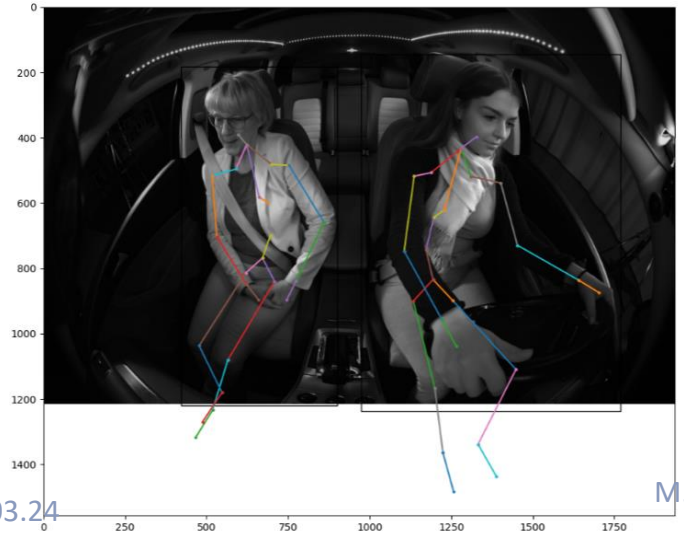
- Computer Vision Group @ RWTH Aachen University
- Estimating Poses of Humans from Images

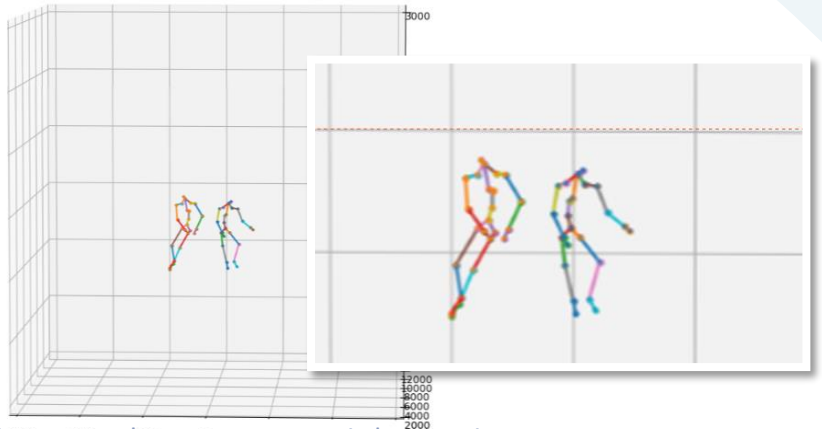
→ Processing images using Convolutional Neural Networks





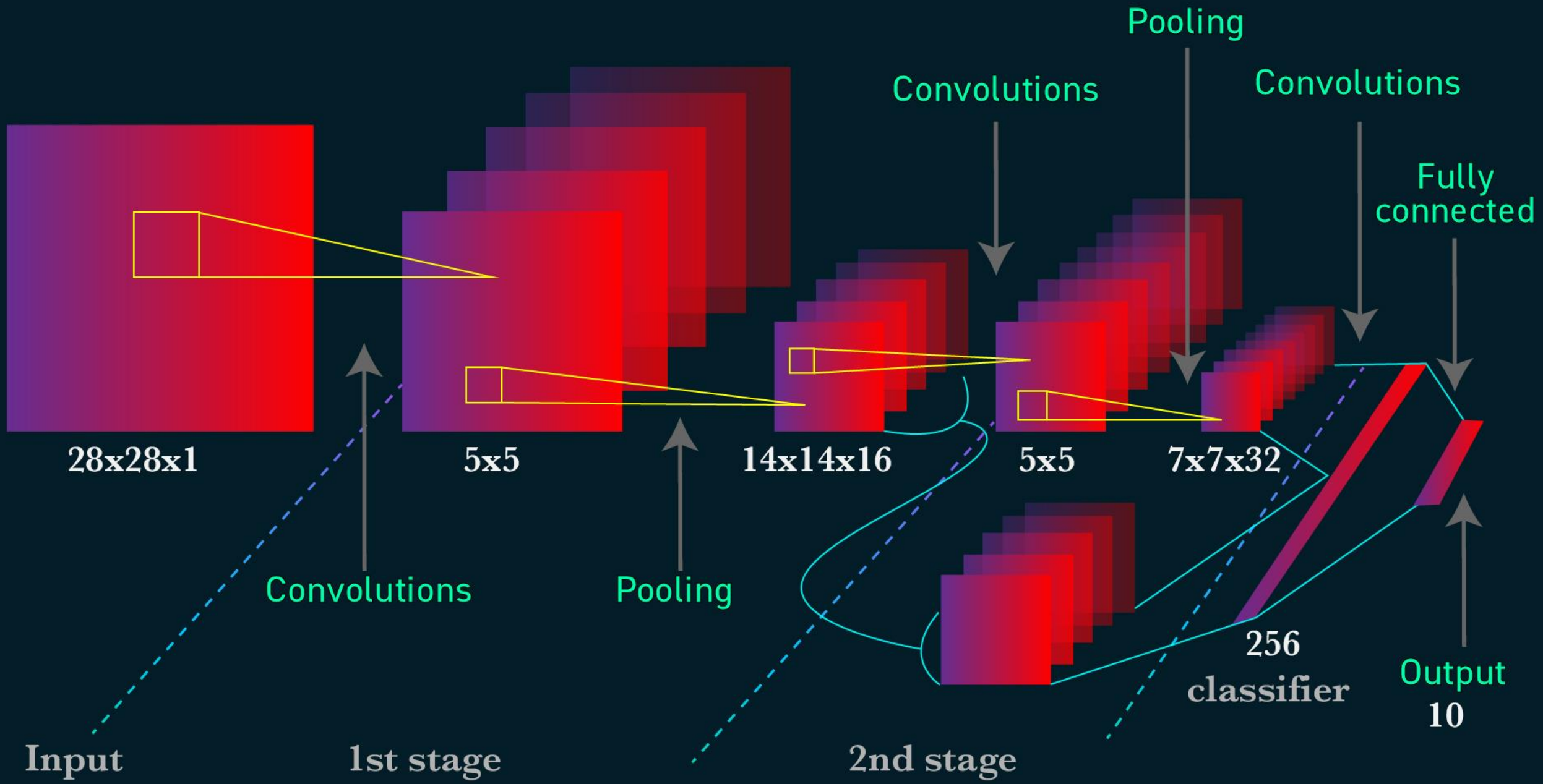






RWTH AACHEN UNIVERSITY

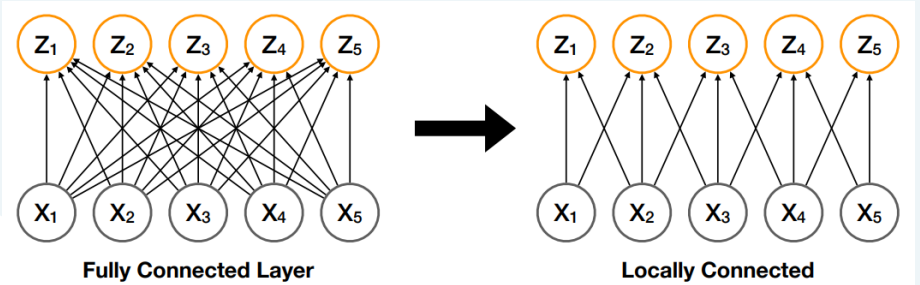
Convolutional Neural Networks - Recap


Pooling
Convolutions
Convolutions
Fully
connected
28x28x1
5x5
14x14x16
5x5
7x7x32
Convolutions
Pooling
256
classifier
Output
10
Input
1st stage
2nd stage

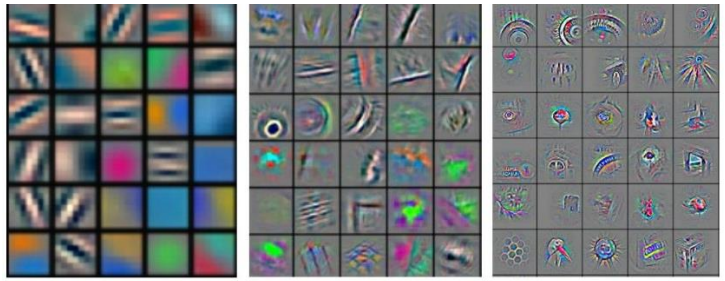
What might be hard to understand?

Images as input

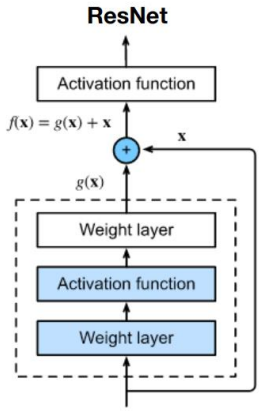
Concept of channels



Reasons for parameter reduction



What filters actually learn



Issues during the training process

Regularization & training tricks

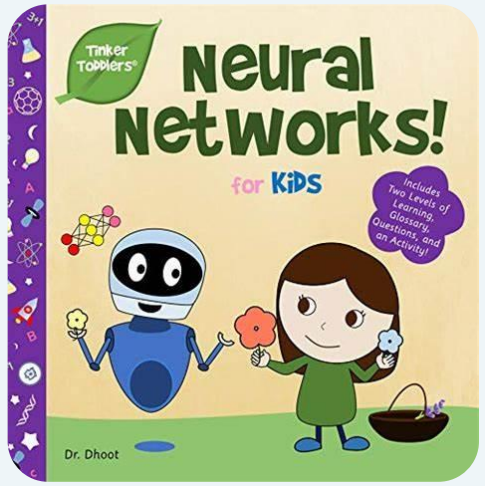
How to set hyperparameters?

What might be hard to understand?

Image: michaelfxu.com

How can we teach CNN basics?

Neural networks for Kids



LeNet - Wikipedia

Databricks Blog

```python
import torch
import torch.nn as nn

class TwoLayerNet(nn.Module):
    """
    In the constructor we instantiate two nn.Linear modules and assign them as
    member variables.
    """
    def __init__(self, input_size, hidden_layers, output_size):
        super(TwoLayerNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_layers)
        self.relu = nn.ReLU()
        self.l2 = nn.Linear(hidden_layers, output_size)

    def forward(self, x):
        """
        In the forward function we accept a Tensor of input data and we must return
        a Tensor of output data. We can use Modules defined in the constructor as
        well as arbitrary (differentiable) operations on Tensors.
        """
        y_pred = self.l1(x)
        y_pred = self.relu(y_pred)
        y_pred = self.l2(y_pred)

        return y_pred
```

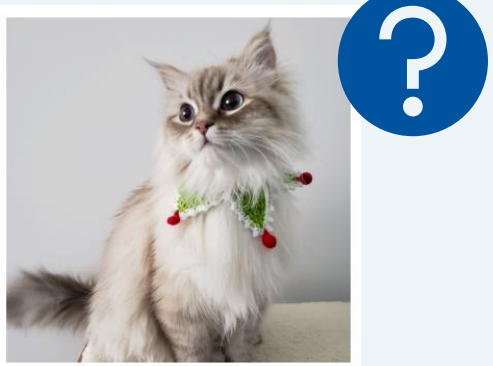
Explain general concepts

Explain common architectures

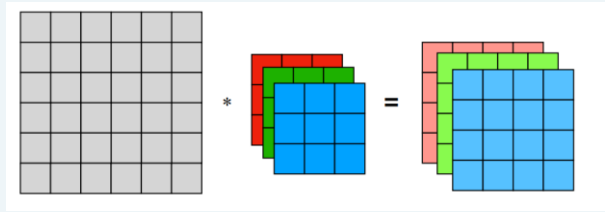
Hands-on coding & calculating exercies


RWTH AACHEN UNIVERSITY
Exemplaric Lecture

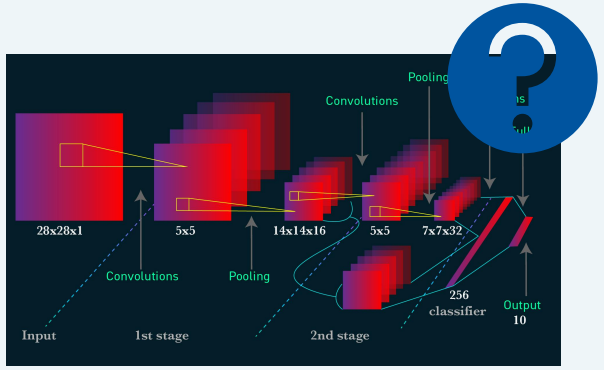
What we'll learn today



What makes image processing challenging?



How do convolutional filters work?



What do we need to build a Convolutional NN?

Overview Slide

Break down what you plan to tackle:
What do you propose?
Why should they care?

What makes image processing challenging?

What we see:

How its encoded

Colours are encoded into RGB components.

Input size [pxl]:	32 x 32
Number of input *channels*:	3
Notation:	32 x 32 x 3

Channels

What makes image processing challenging?

What we see:

How its encoded

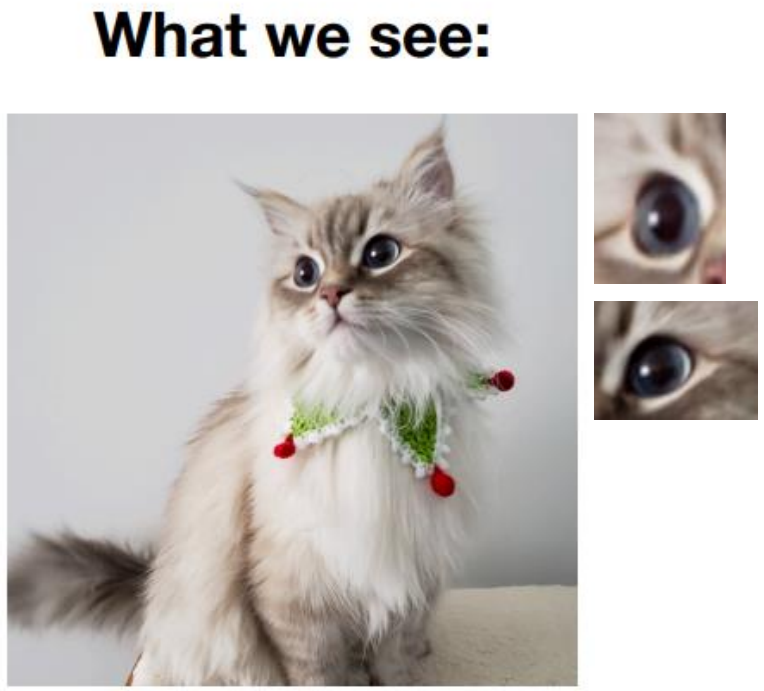
5	10	2	4	2	0
3	6	2	0	1	3
0	7	10	4	8	0
11	10	8	0	3	7
10	4	8	6	0	4
5	12	11	0	2	0

5	6	2	3	2	0
5	6	2	0	1	3
0	5	12	2	5	6
2	2	8	0	3	7
0	4	5	10	0	4
4	1	4	0	2	9

6	10	2	4	2	2
3	6	12	0	1	6
0	5	2	4	5	4
11	13	8	1	3	7
9	4	4	6	0	9
5	12	5	8	2	9

Images are matrices.

What makes image processing challenging?



Patterns might repeat among the images.

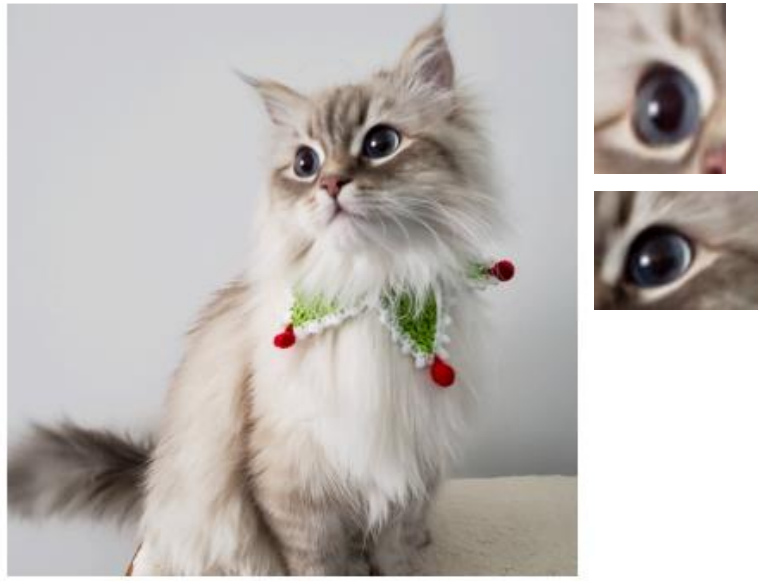
Difficulty 1: MLPs do not consider spatial data structure

- MLPs treat each feature **independently** without considering the spatial structure of the data
- **Spatial arrangement of pixels** is crucial for understanding the image

Argue why MLPs are not perfect

What makes image processing challenging?

What we see:



Patterns might repeat among the images.

Difficulty 2: Lack of translational invariance

- MLPs are **not robust to shifts** in the image
- If an object is in a **different location**, the MLP may have difficulty recognizing it
- MLP might need to relearn for every feature position

Argue why MLPs are not perfect

What makes image processing challenging?

No. of input values is quite high, especially for high-resolution images.

Input size [pxl]: 32 x 32
Number of input *channels*: 3
Notation: 32 x 32 x 3

32 * 32 * 3 = 3072 input values

How can we process this using a MLP?

Make an example for parameter

What makes image processing challenging?

How can we process this using a MLP?

10 types of cats & use 100 neurons

Model:	MLP
Input:	32 x 32 x 3
Hidden Layer:	100 nodes
Output Classes:	10

- Each neuron in the **hidden layer** has
  - 3072 weights (one per input)
  - a bias term

3072 * 100 + 100 parameters

- Each neuron in the **output layer** has
  - 100 weights (one per neuron in the hidden layer)
  - a bias term

100 * 10 + 10 parameters

Total number of parameters: (3072×100+100)+(100×10+10) = **308,310**

Make an example for parameter

What makes image processing challenging?

How can we process this using a MLP?

10 types of cats & use 100 neurons

Difficulty 3: High number of parameters

- Image input vectors can be very large
- Large number of parameters in the network
- Risk of
  - overfitting
  - slower training times

RWTH AACHEN UNIVERSITY

Caution:
Make sure people know what learnable „***parameters***“ in NNs are i.e. that here you are referring to the „***weights***“.

What makes image processing challenging?

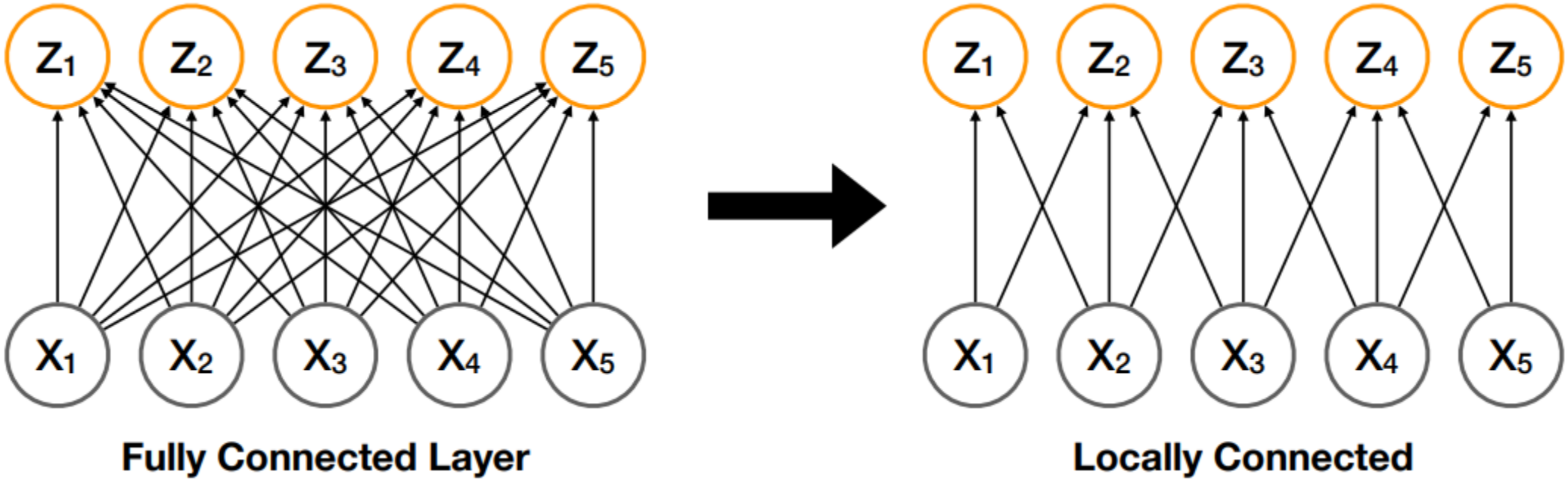
Difficulty 1: MLPs do not consider spatial data structure
Difficulty 2: Lack of translational invariance
Difficulty 3: High number of parameters

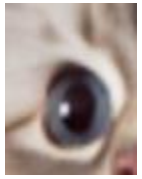
What do we want/need?

1. **Local connections**: Connections only between neighbouring inputs (pxls)
2. **Translational invariance**:
   - Shifted input results in similar output
   - Parameter sharing

What makes image processing challenging?

Solution: Local Connections

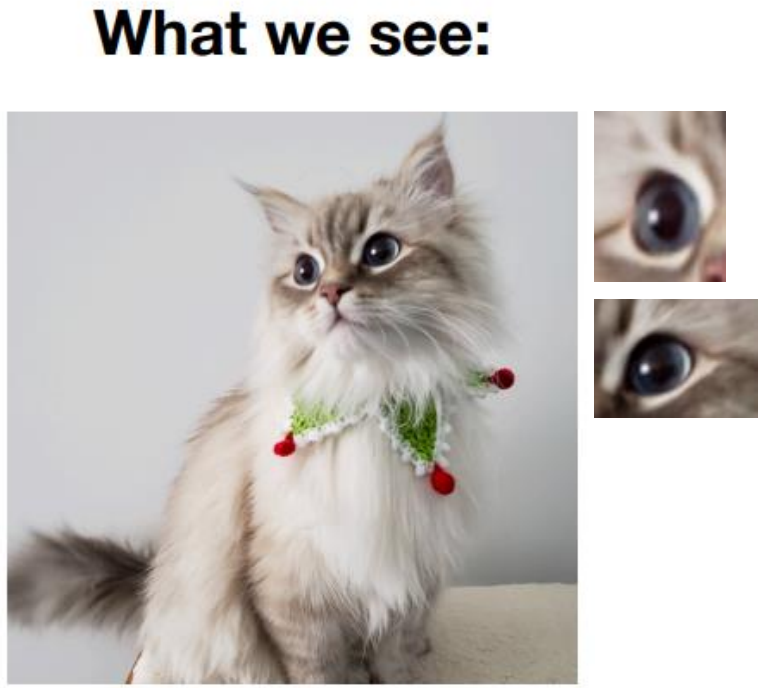


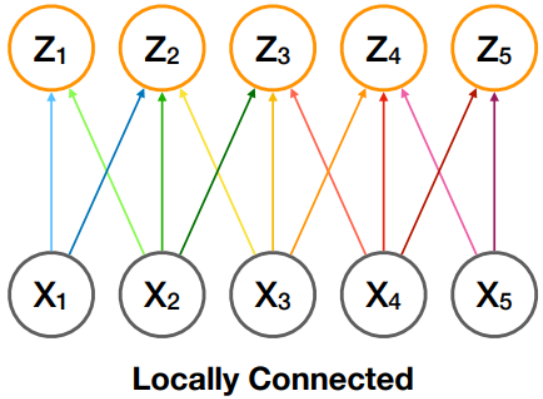


- Image features are (typically) limited to a set of neighboring pixels
- Not entire image needs to be ‘seen’ at once

What makes image processing challenging?

Solution: Parameter Sharing





- Same features can appear at various positions
- Weights that detect features can be repeatedly applied

-> reduces total no. of parameters

This can be achived using convolutional filters.

What makes image processing challenging?

Difficulty 1: MLPs do not consider spatial data structure
Difficulty 2: Lack of translational invariance
Difficulty 3: High number of parameters

CNNs can solve these problems!

CNNs…

- capture spatial structures of images
- are translationally equivariant
- have fewer parameters than MLPs

How do convolutional filters work?

Convolution:
Mathematical operation combining two functions to generate a third function.

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau$$

To convolve a kernel with an input signal:
flip the signal, move to the desired time,
and accumulate every interaction with the kernel

Image: cloudzsexy.com

Depending on the audience you might want to introduce some math.

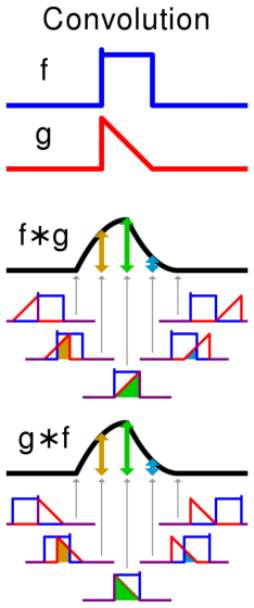
How do convolutional filters work?

Convolution:
Mathematical operation combining two functions to generate a third function.

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau$$

To convolve a kernel with an input signal:
flip the signal, move to the desired time,
and accumulate every interaction with the kernel

Image: cloudzsexy.com



- Continuous convolution for real-valued functions f and g $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$
- Our image inputs are discrete

➡ Discrete formulation $(f * g)(n) = \sum_{m=-\infty}^{\infty} f(m) g(n - m)$

- Two dimensional case with Image I and Kernel K:

$$(I * K)(i, j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m, n) K(i - m, j - n) = (K * I)(i, j)$$

How do convolutional filters work?

Image I

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

*

Kernel K

0	1	0
1	2	1
0	1	0

=

I ∗ K

$$(I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)K(i-m,j-n) = (K * I)(i,j)$$

Make sure to demonstrate the filter operation on the blackbord

How do convolutional filters work?

Image I

1_0	1_1	1_0	0	0
0_1	1_2	1_1	1	0
0_0	0_1	1_0	1	1
0	0	1	1	0
0	1	1	0	0

$*$

Kernel K

0	1	0
1	2	1
0	1	0

$=$

I $*$ K

4		

$$(I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)K(i-m,j-n) = (K * I)(i,j)$$

Make sure to demonstrate the filter operation on the blackbord

How do convolutional filters work?

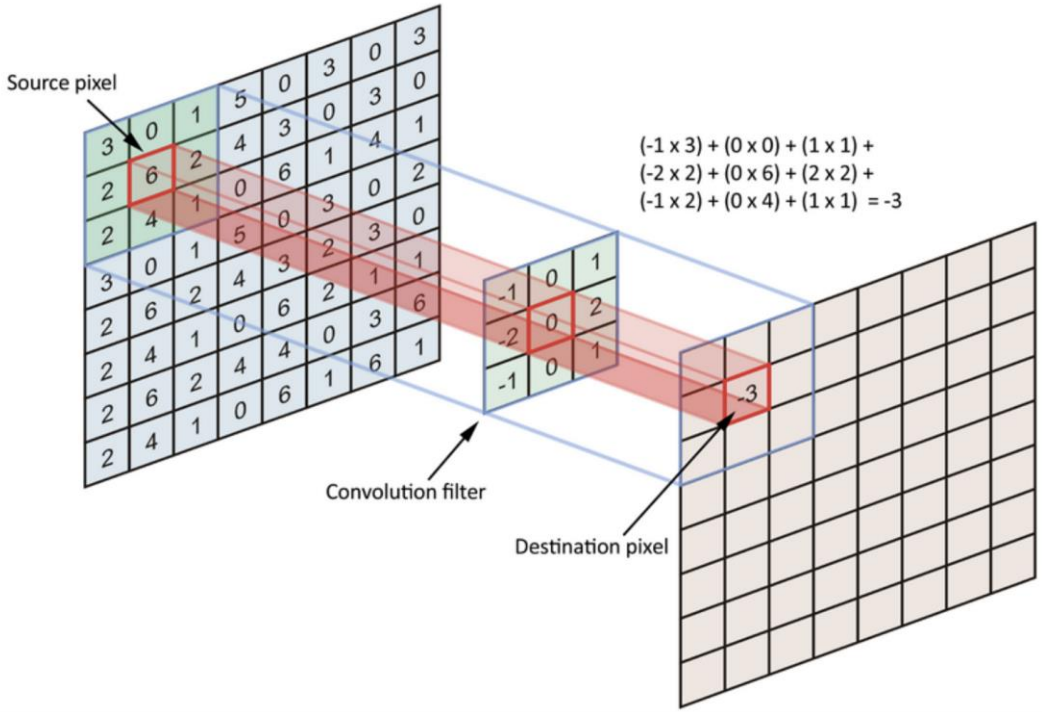


Image: Simple Introduction to Convolutional Neural Networks | by Matthew Stewart, PhD | Towards Data Science

And show 3D images to make it even clearer.

How do convolutional filters work?

Convolutional Filters:

- Used to **extract image features**
  i.e. recognize specific patterns in the input
  -> can be edges, textures or structural elements.
- Filter **weights are learned** during training

Early layers learn **simple features** like edges, later layers combine them into **more complex features** like shapes or objects.

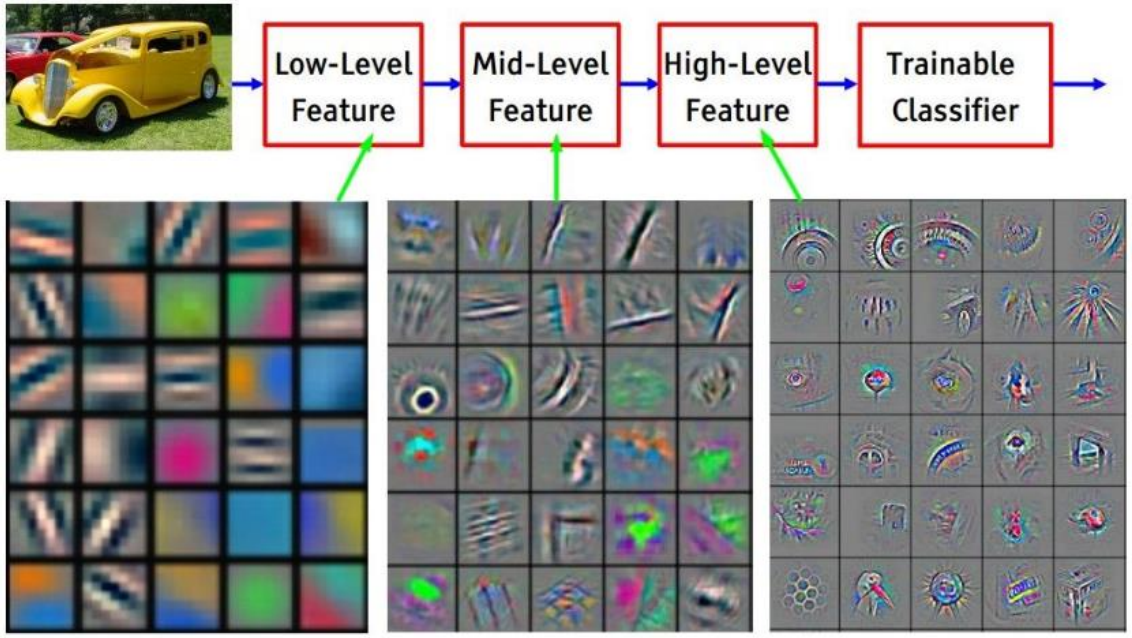
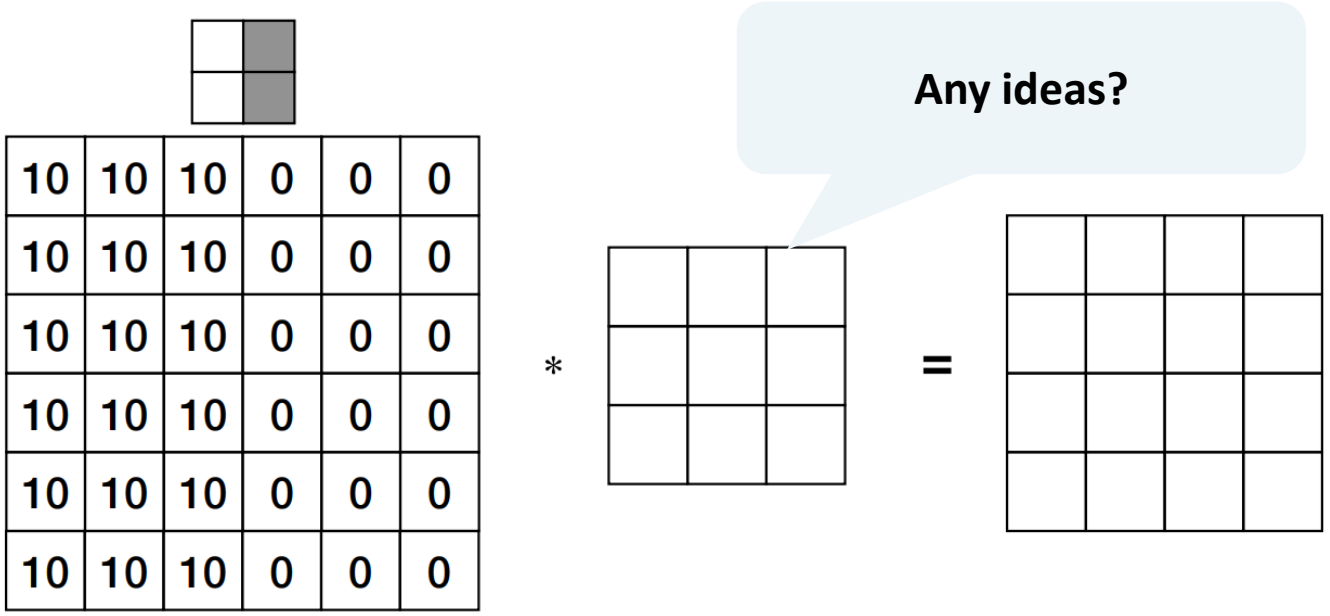


Image: tpsearchtool.com

What do filters learn?

How do convolutional filters work?

Application: Edge detection



Demonstrate examples of classic filter operations

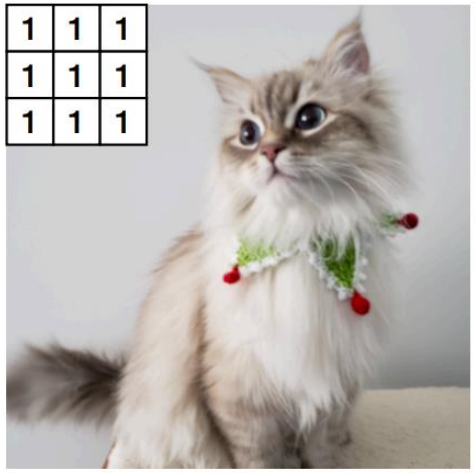
How do convolutional filters work?

Application: Blurring and Sharpening Images

Original

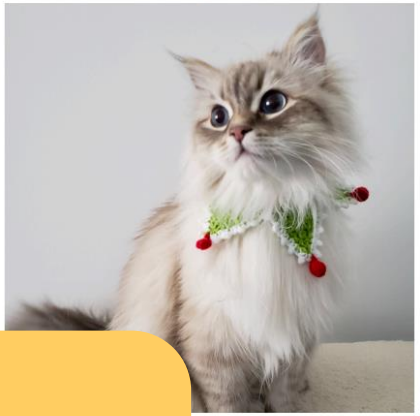
Blur

1	1	1
1	1	1
1	1	1



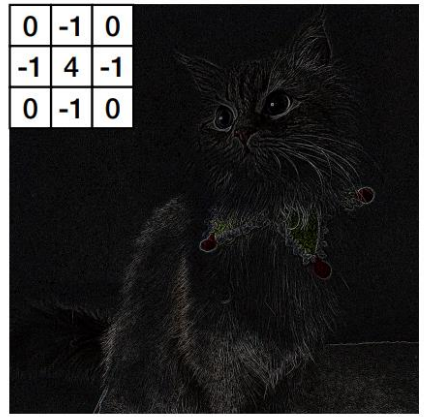
There are many specific filters for specific purposes.

Original



Multi Edge

0	-1	0
-1	4	-1
0	-1	0



Sharpen

0	-1	0
-1	5	-1
0	-1	0

Demonstrate examples of classic filter operations.

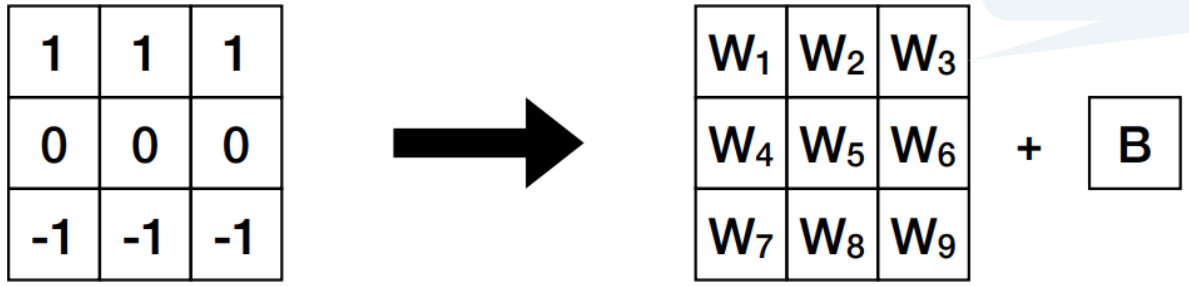
How do convolutional filters work?

Application: Blurring and Sharpening Images

There are many specific filters for specific purposes.

How can we generalize?

This is what is learned / updated in our network!



Idea: Replace kernel entries (i.e. hard-coded filter values) with trainable weights + bias term
→ Convolutional Neural Network Layer

Now explain that these filter values are

What do we need to build a Convolutional NN?

We can build it ourselves or use a ML library.

Popular:
- Tensorflow
- PyTorch

Building block-like creation of networks.

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

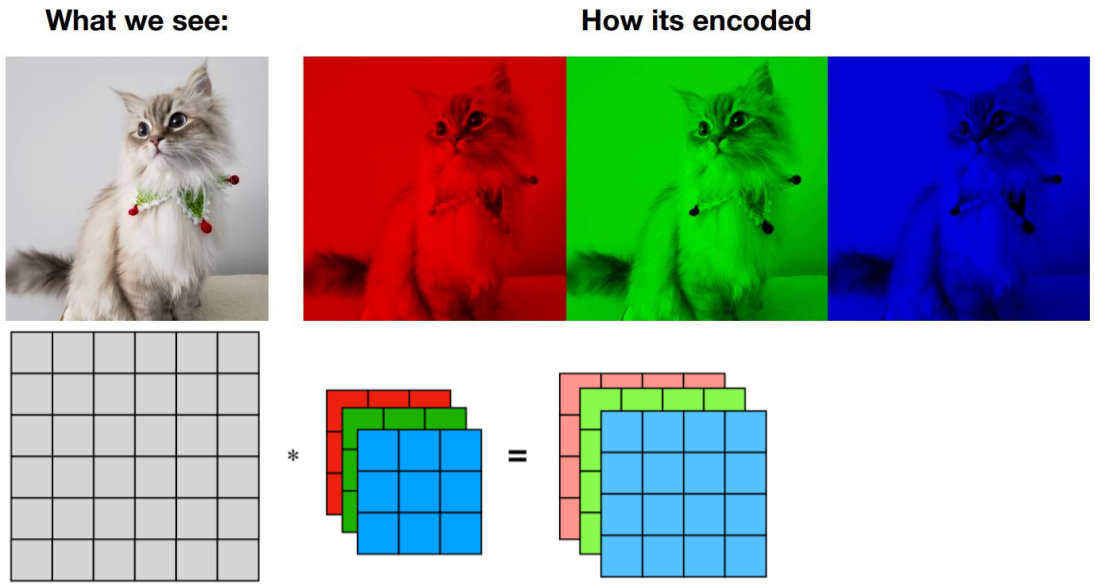
Image: Trainin

Make sure students understand common terminology.

What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Channels



One filter may not be sufficient to extract all important features! Example: RGB images

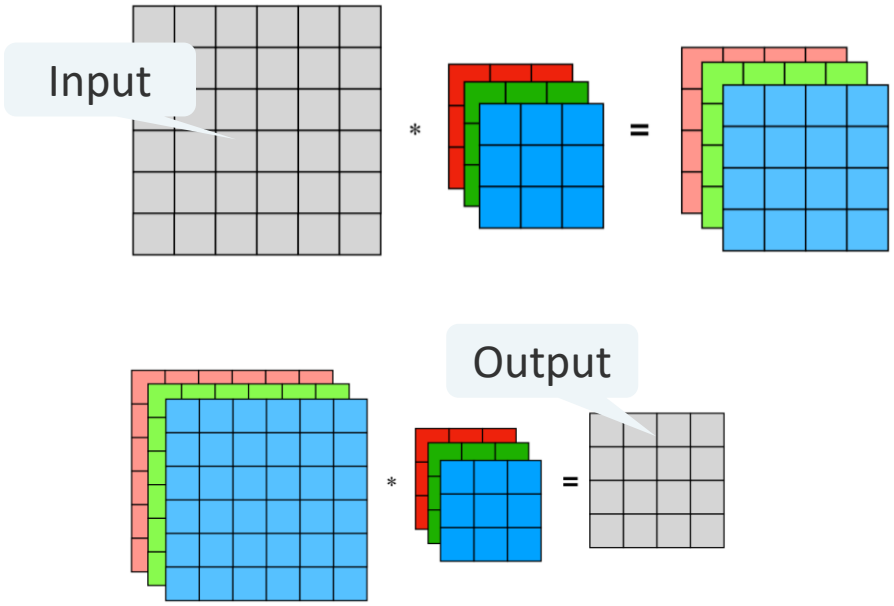
- Use multiple filters for multiple output channels/feature maps
- Each filter has independent weights

Make sure students understand common terminology.

What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Channels



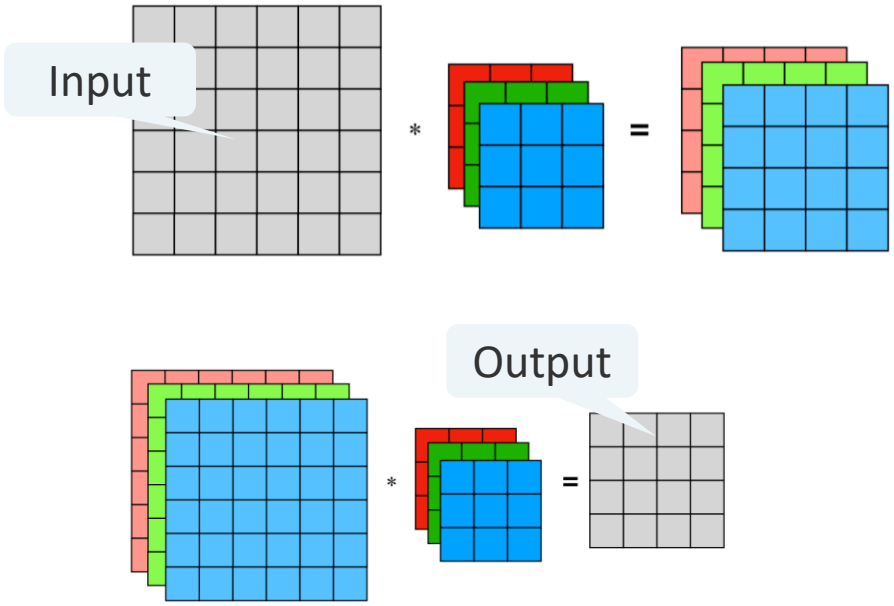
One filter per input channel, sum over each convolution result

We typically only want a single output channel, containing information from all inputs

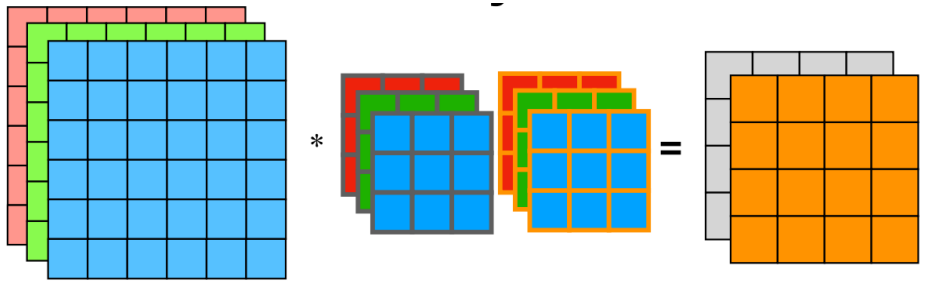
What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Channels



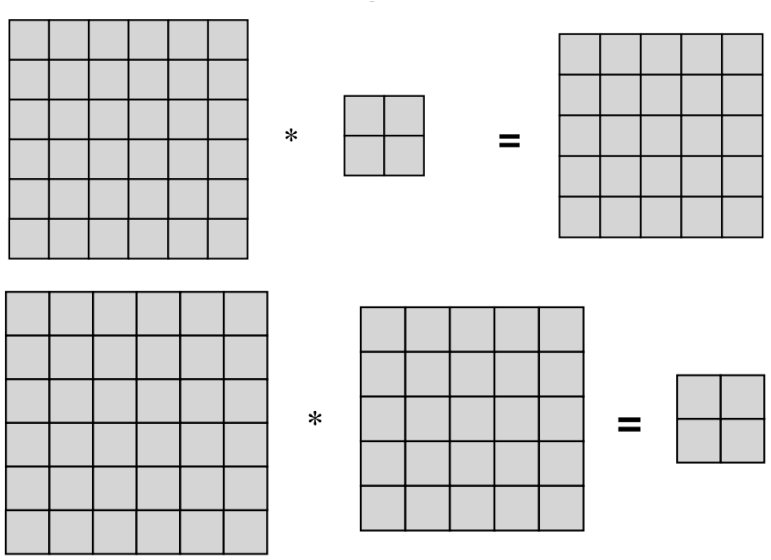
However, multiple input and output channels are also possible!



What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Kernel Size / Filter Size:



Kernel size dictates “how far” pixels are connected (in their neighborhood):

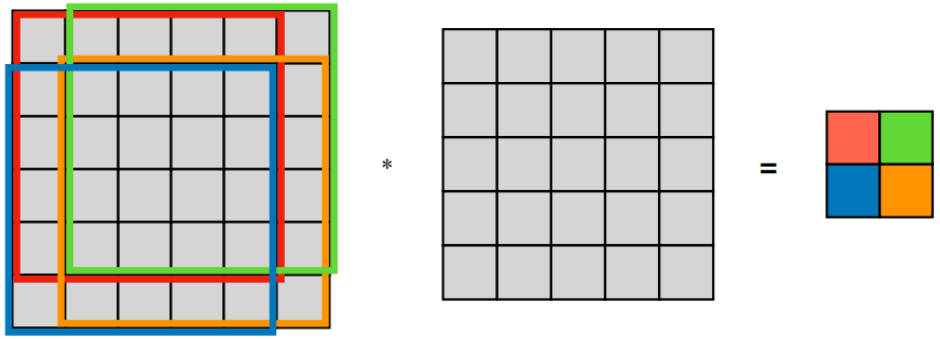
- Too small kernel: insufficient to cover interesting features
- Too large kernel: results in too many weights

Make sure students understand common terminology.

What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

The Shrinking of Convolution Outputs



Kernel needs to overlap fully with input image

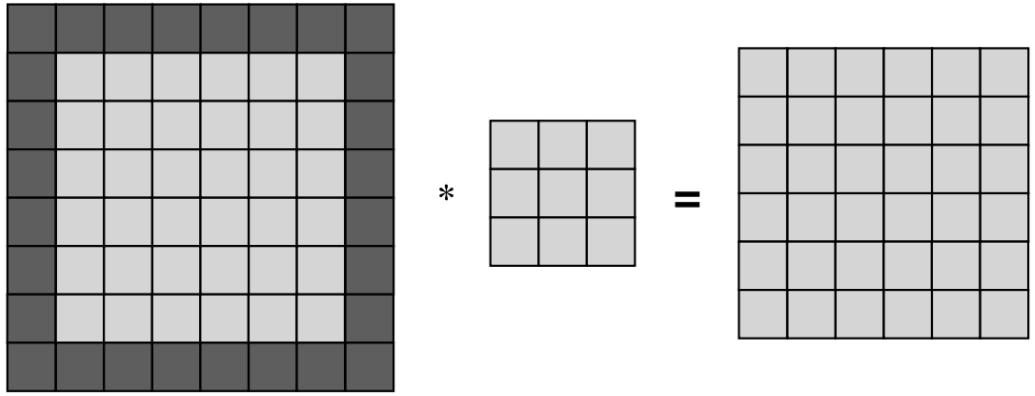
For kernel size > 1:
reduces the output size (compared to input)
→ default setting (“*valid padding*”)

For understanding padding it helps to see an image „shrink“.

What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Padding



'Same' padding: pads such that input and output are same size

Kernel "cannot" be applied to pixels at the edges so that our output images preserves its image size

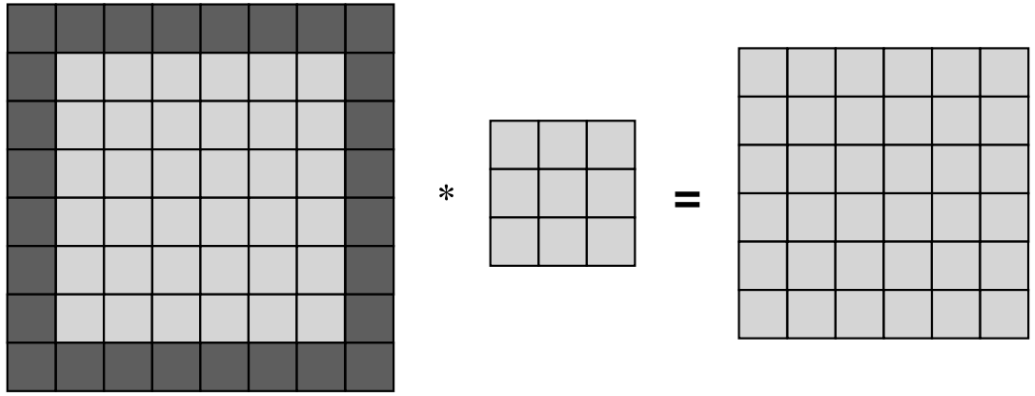
-> Padding is a solution for artificially enlarging our images, so that our images do not "shrink" during convolution

What do we need to build a Convolutional NN?

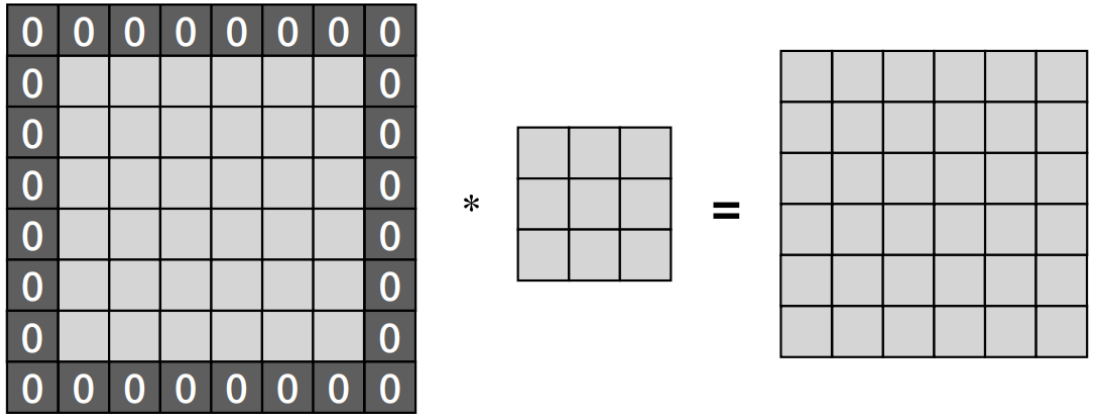
Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

There are many more padding methods. Guide to Different Padding Methods for CNN Models - (analyticsindiamag.com)"

Padding



‘Same’ padding: pads such that input and output are same size



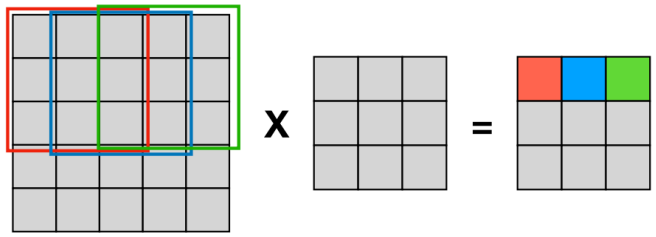
‘Zero’ padding: pads given area using 0 as a fill value
Can have ‘Zero’ (value) and ’Same’ (shape) padding

What do we need to build a Convolutional NN?

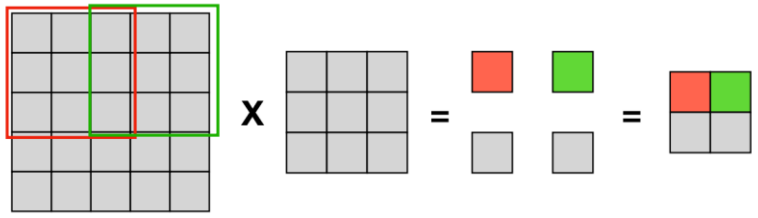
Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Stride

Stride = 1: Move kernel by 1 pixel



Stride = 2: Move kernel by 2 pixel



Strides describes how the convolutional kernel moves (or “jumps”) in each step

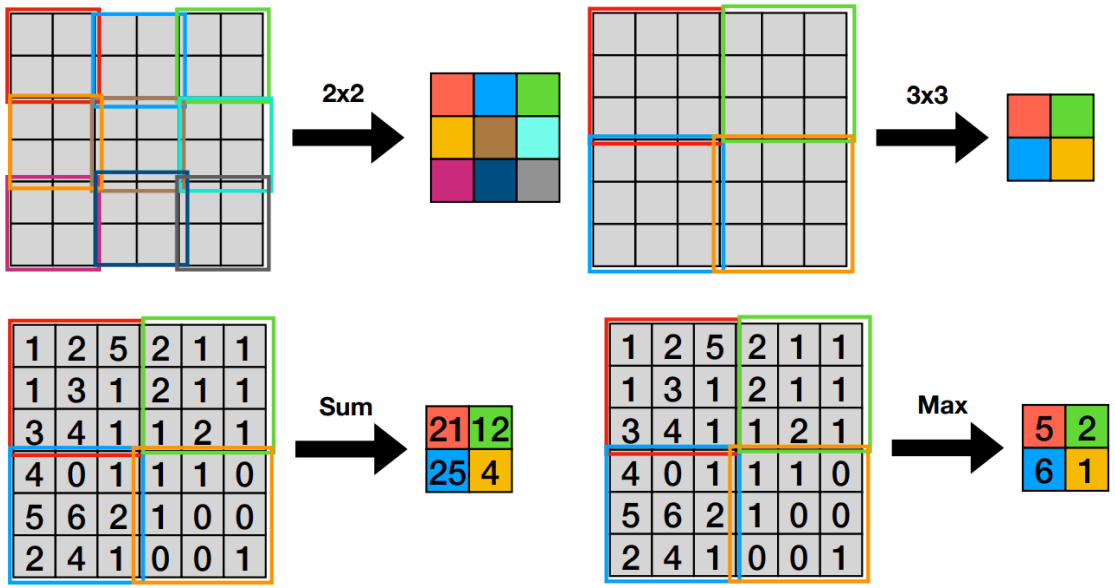
Strides > 1 act as *downsampling.*

Let them calculate examples on a sheet of paper.

What do we need to build a Convolutional NN?

Important technical details for a Convolutional Layer:
Channels • Kernel size • Padding • Strides

Pooling



Reducing the image dimensionality is often required. Options:

a) Stride convolutions
b) Pooling

Pooling:
Combines information of several pixels into one pixel
-> for ex. by taking the **sum**, **mean** or **max**

Again: Pooling kernel size influences the output size, i.e. how much the image is downsampled

What do we need to build a Convolutional NN?

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,
dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)
```

- Readily implemented in most ML libraries
- Important technical details summary:
  - **Channels:** number of input and output channels
  - **Kernel size:** choose appropriate to feature size
  - **Padding:** adds ‘empty’ pixels to manipulate output size
  - **Strides:** ‘step size’, allows for downsampling

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

Image: Trainin

Always come back to your overview slide …

(Historically) Popular Network Architectures

Many of them were created for the…

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

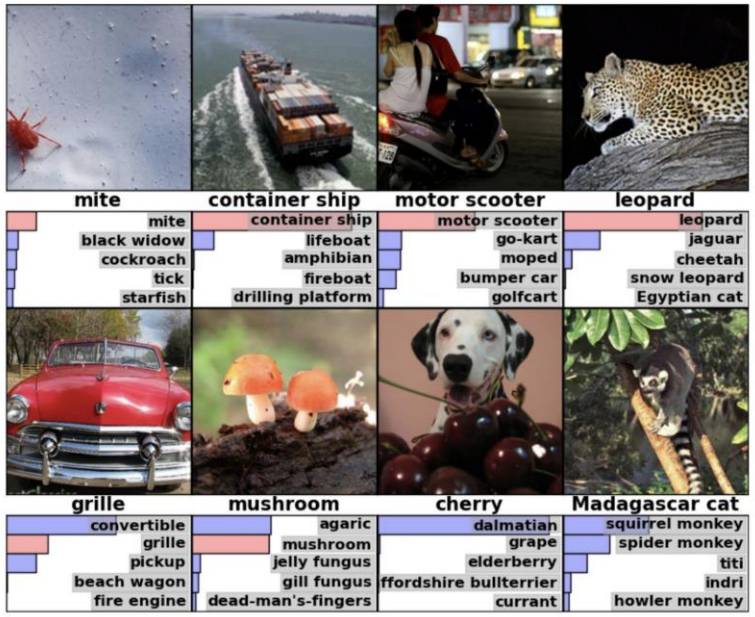


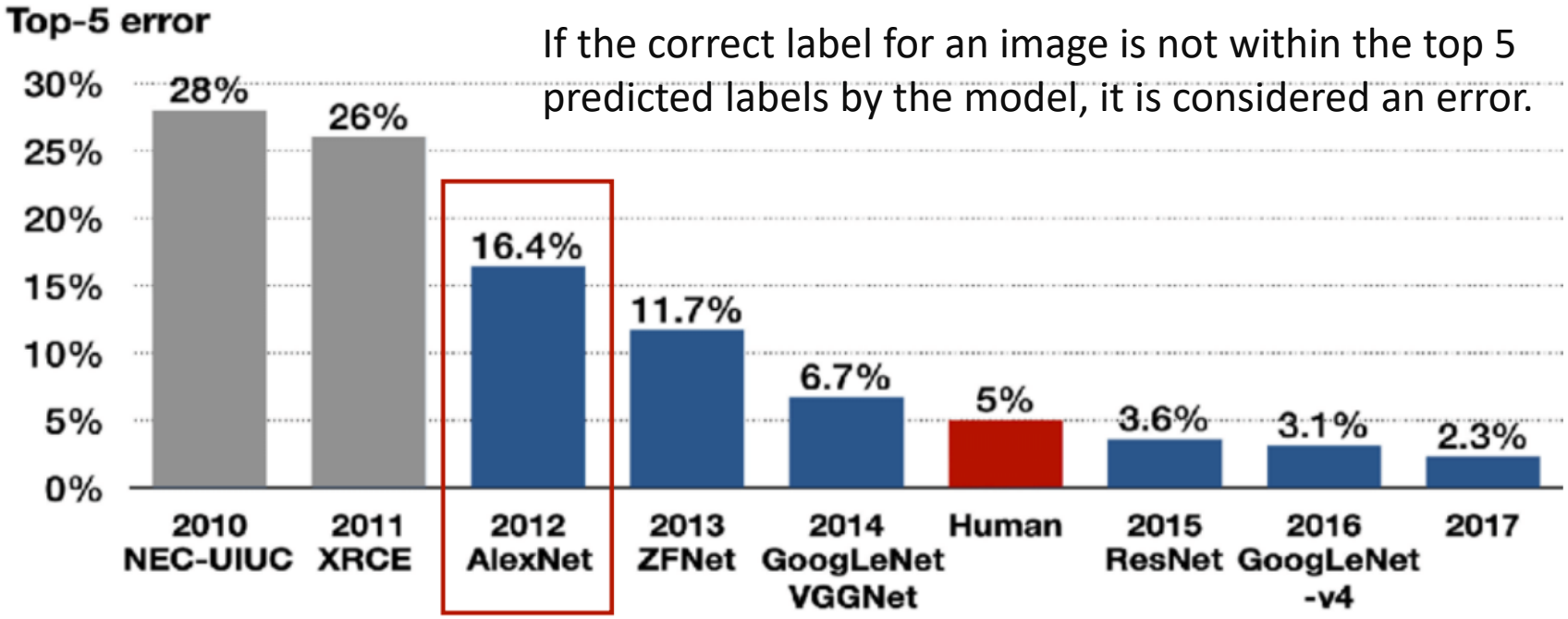
Image: NLP's ImageNet moment has arrived (thegradient.pub)

Image: Meme Creator

Introduce common benchmarks.

(Historically) Popular Network Architectures

Image Net Challenge



If the correct label for an image is not within the top 5 predicted labels by the model, it is considered an error.

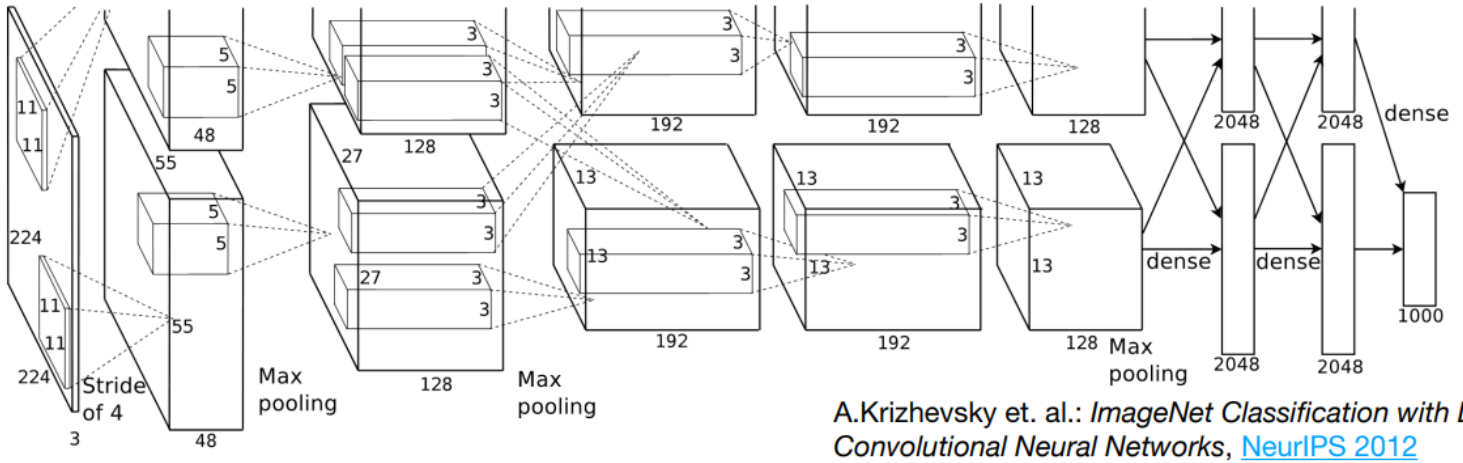
Dae-Young Kang et. al.: *Application of Deep Learning in Dentistry and Implantology*, 10.32542

Introduce common benchmarks.

(Historically) Popular Network Architectures

AlexNet

- Archetypal for most convolutional networks
  - Convolutional segment
  - Dense segment

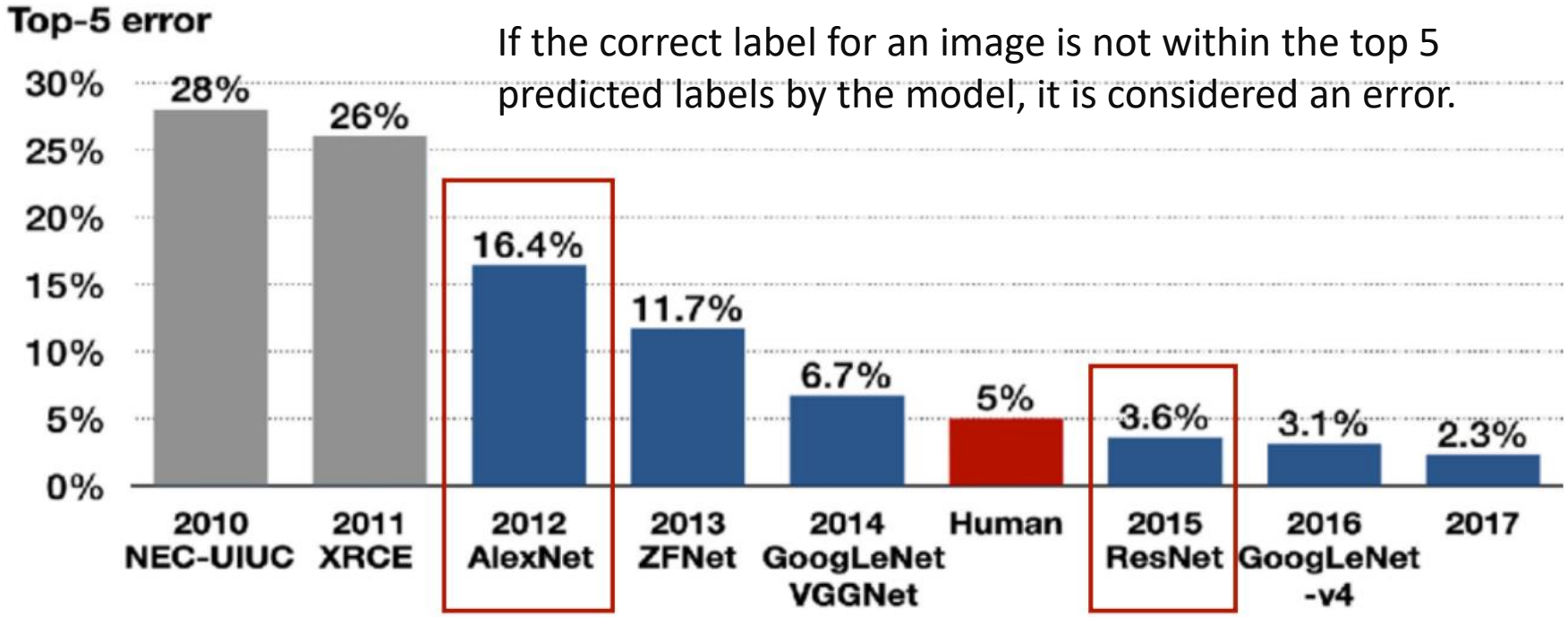


A.Krizhevsky et. al.: *ImageNet Classification with Deep Convolutional Neural Networks*, NeurIPS 2012

Give an overview of well-known architectures

(Historically) Popular Network Architectures

Image Net Challenge



If the correct label for an image is not within the top 5 predicted labels by the model, it is considered an error.

Dae-Young Kang et. al.: *Application of Deep Learning in Dentistry and Implantology*, 10.32542

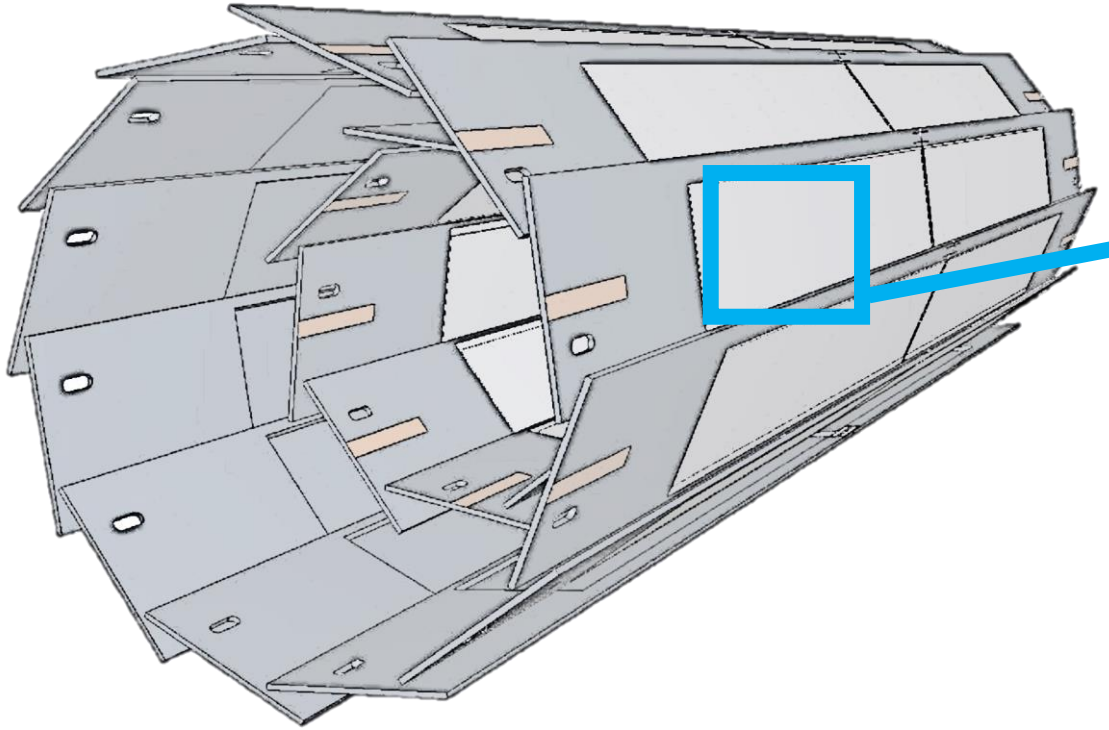
..but state that they are not SOTA anymore

RWTH AACHEN UNIVERSITY

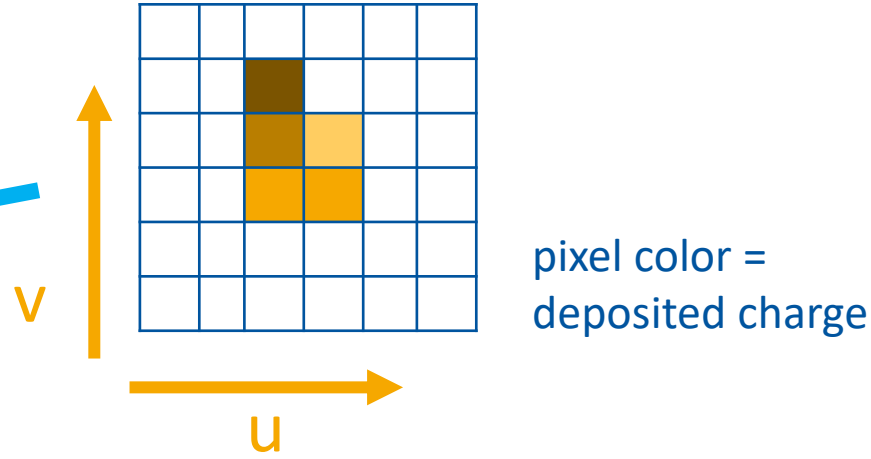
Make sure they understand that **deeper is not always better**!

(Historically) Popular Network Architectures

…and why they are sometimes not well-suited.



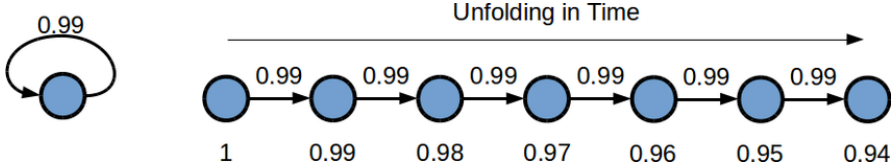
belle2.desy.de



If we apply a **deep architecture**, small images might **disappear**!

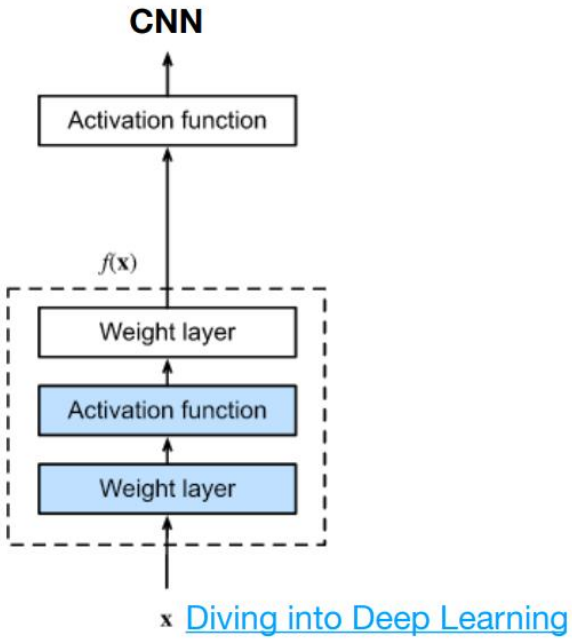
(Historically) Popular Network Architectures

ResNet & Vanishing Gradient



researchgate.net

- Standard CNNs have limited depth, too deep
- ➡ Vanishing gradients
  - Negative weights
  - Negative output
  - ReLU maps to zero
  - No gradient
- More likely to occur for deep networks



Diving into Deep Learning

Vanishing Gradient Problem

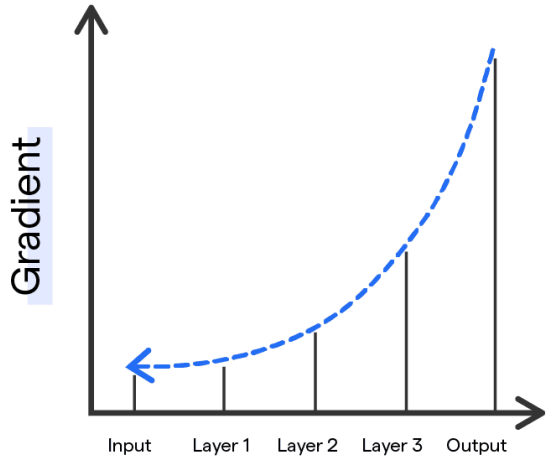


Image: BotPenguin

Do not forget the vanishing gradient problem!

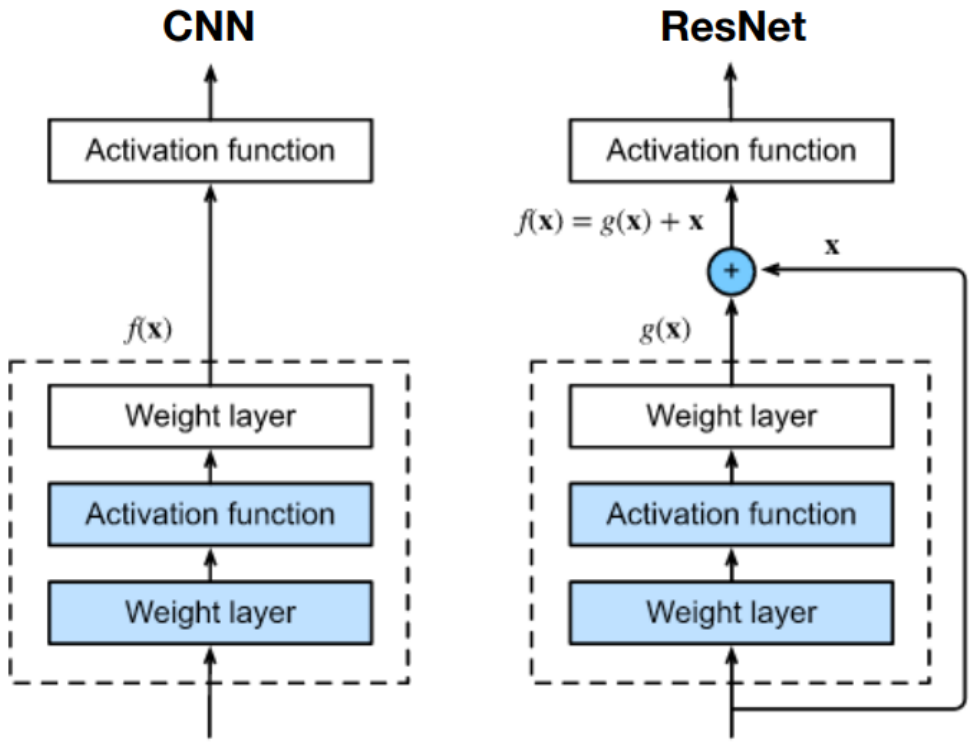
(Historically) Popular Network Architectures

ResNet

- Standard CNNs have limited depth, too deep

➡ Vanishing gradients

- Solution:
  - Residual/skip connections
  - Gradients can propagate through network easily
  - Conv. layers only learn modification to input

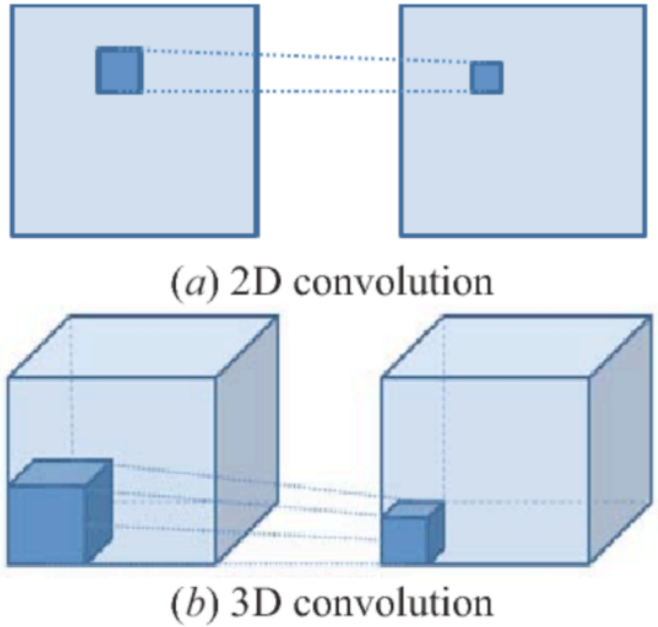


Diving into Deep Learning

Outlook: 3D Convolutions

Convolutions not limited to 2D

- Concept can be extended to 3D data as well
  - 3D output, 3D kernel, 3D strides, ….
  - Volumetric data processing
- Similar for 1D
  - Time sequence processing



(*a*) 2D convolution

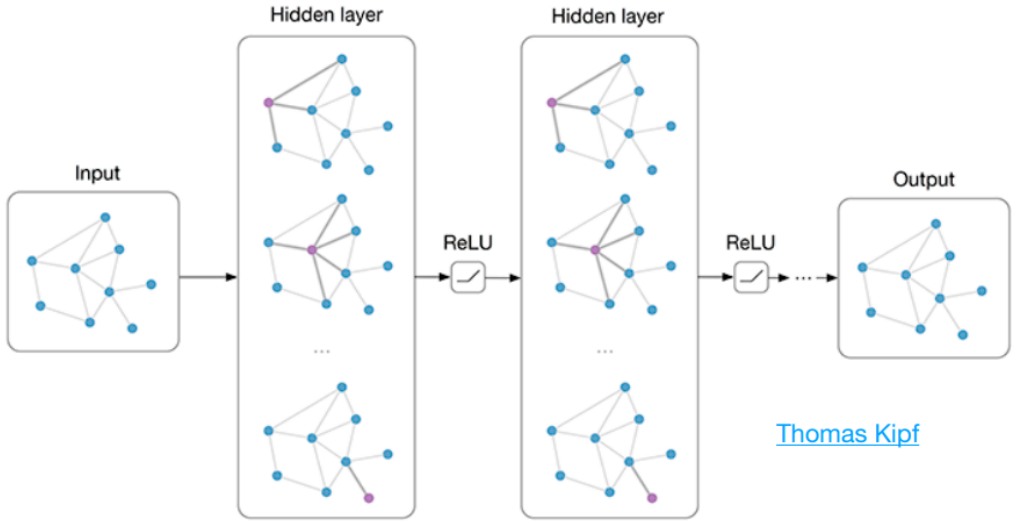
(*b*) 3D convolution

Fan et. al.,Lung nodule detection based on 3D convolutional neural networks, document/88

Do not forget to mention other

Outlook: Graph Convolutions

- Graph CNNs
- Operate on graphs with arbitrary node positions
- Do not require fixed structures like CNNs
- More in advanced course

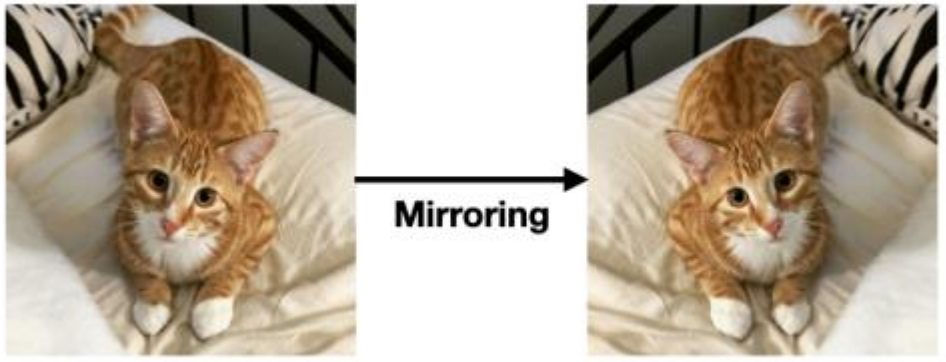


Thomas Kipf

Do not forget to mention other variants of CNNs


RWTH AACHEN UNIVERSITY
Explaining the Training Process

What do we need to train a Convolutional NN?



Data Augmentation:
Mirroring, Flippig, Rotating, Cropping of our images etc.
-> artificial increase of available data
-> less overfitting

What do we need to train a Convolutional NN?

Hyperparameters?

Hyperparameter	Range	Optimal Value
learning rate	[0.001, 0.01, 0.1, 0.2]	0.01
batch size	[5, 10, 20]	20
epochs	[8, 16, 32]	32
optimization algorithm	[SGD, RMSprop, Adam]	SGD

The range and optimal values of hyperparameters for CNN. | Download Scientific Diagram (researchgate.net)

- Fitting a model is an **optimization problem**
- We **cannot test all possbilities**
- We rely on **heuristics** for choosing them

What do we need to train a Convolutional NN?

Hyperparameter	Range	Optimal Value
learning rate	[0.001, 0.01, 0.1, 0.2]	0.01
batch size	[5, 10, 20]	20
epochs	[8, 16, 32]	32
optimization algorithm	[SGD, RMSprop, Adam]	~~SGD~~

Explain tradeoff between training time and quality of the results.

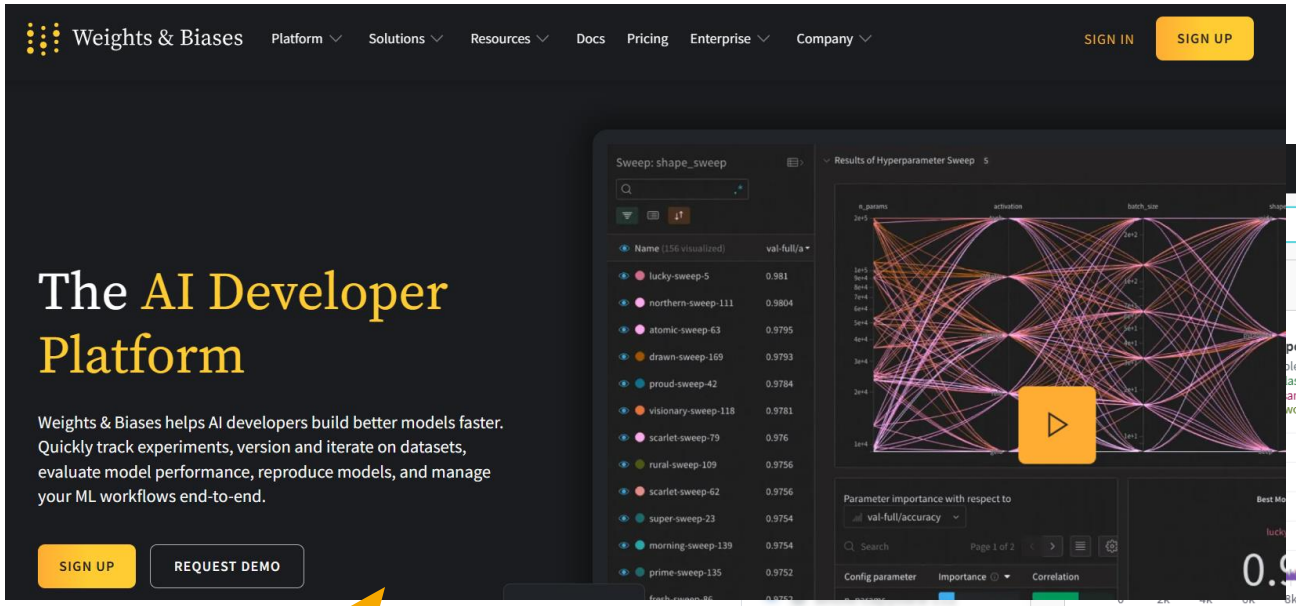
Adam Optimizer is often a good choice.

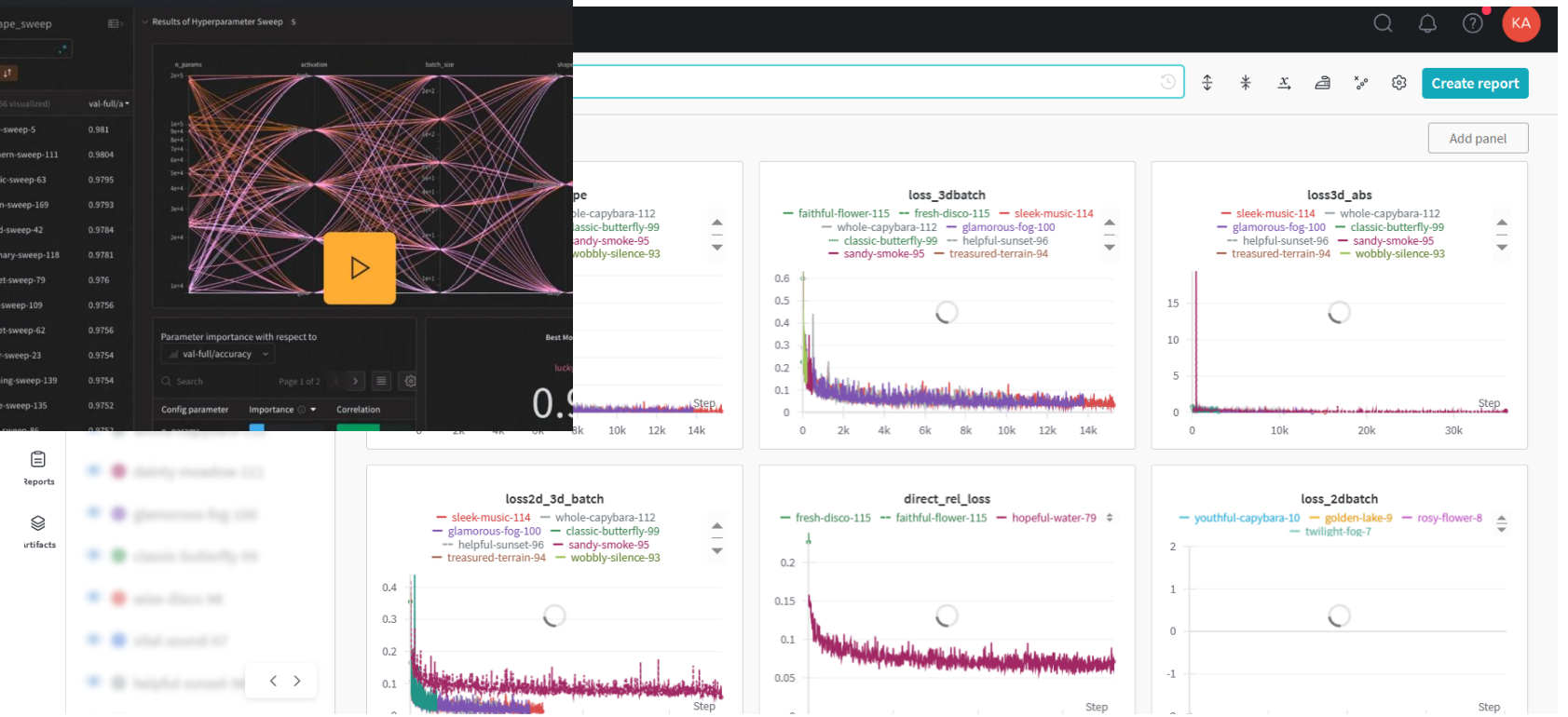
The range and optimal values of hyperparameters for CNN. | Download Scientific Diagram (researchgate.net)

Image: tomshardware.com

Mention benefits of (Nvidia) GPUs

What do we need to train a Convolutional NN?





Useful visualization tools!

What do we need to train a Convolutional NN? Clean code.

Don‘t forget to comment your code!

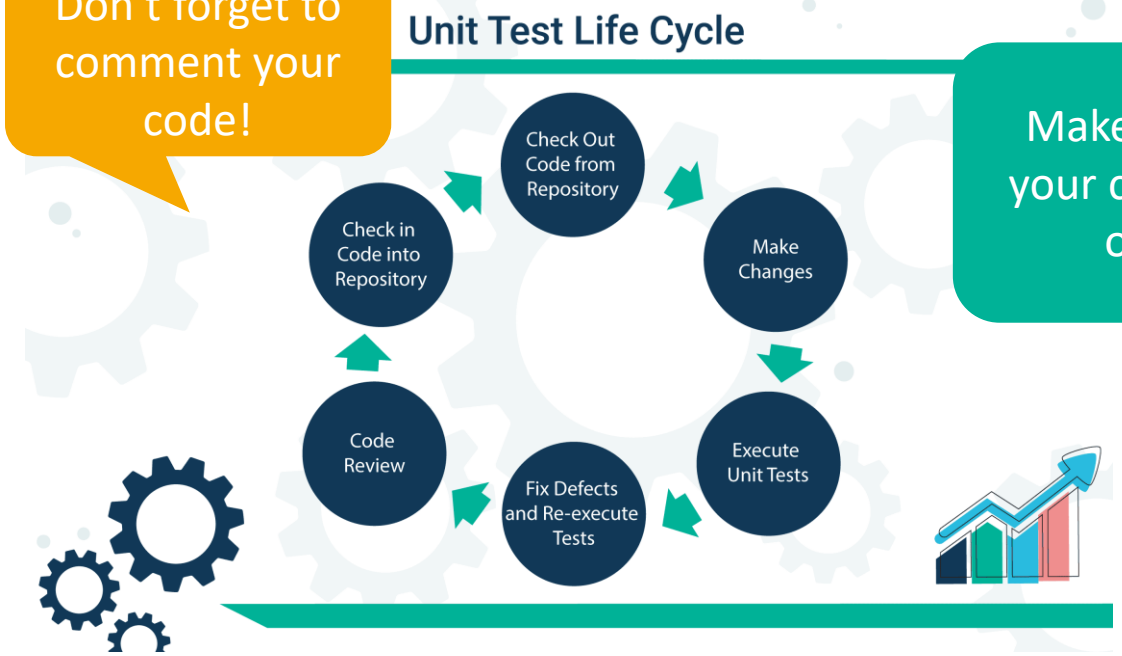
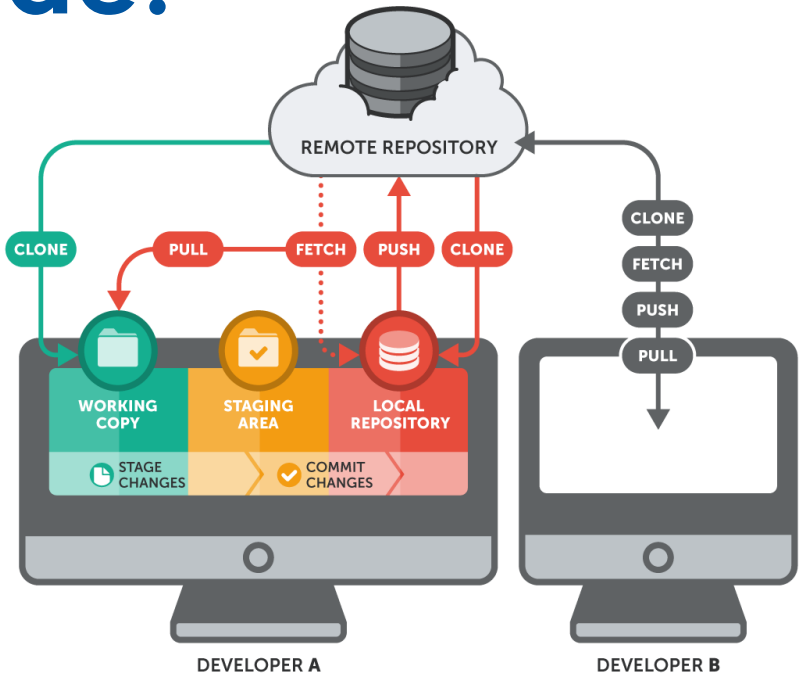


Image: Unit Test: What is, Advantages & How works? - CodenBox AutomationLab

Make sure to keep your code clean and organized.



10 Best Git Tutorials for Beginners [2024 FEB]- Learn Git online | Quick Code (medium.com)

Use unit tests and git for version control.

Possible Exercises

CNN-Theory Tasks

Ask for

- The number of **learnable parameters**
- The **output size** of a convolutional layer
- The influence of **stride/pooling/padding** etc.
- Setting up a **feasible layer stucture**
  (i.e. not too many layers if image is small, add dropouts if no of parameters is high etc.)
- Explanation of **code fragments**
- The influence of **skip connections**
- Benefits of **ReLU**
- Role of **FCN layers** in CNN architectures
- …

Coding Tasks

Refresh Skills
-> provide tutorial

Use **Python**
(Julia and R are harder to debug)

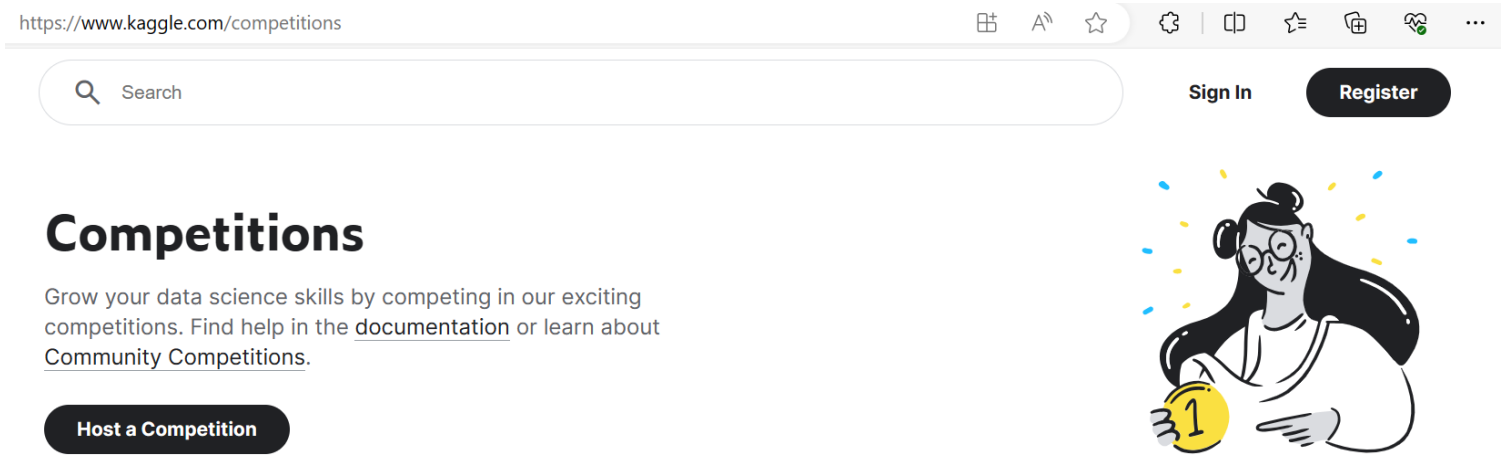
Either **PyTorch**
or Tensorflow

Support
best practices
(Git, Wandb / TensorBoard, Unit Tests)

Coding Tasks

- Start with simple **architectures** like LeNet
- **Gradually increase complexity** by adding argumentation, regularization etc.
- Make use of **well-known baselines** (ImageNet etc.)

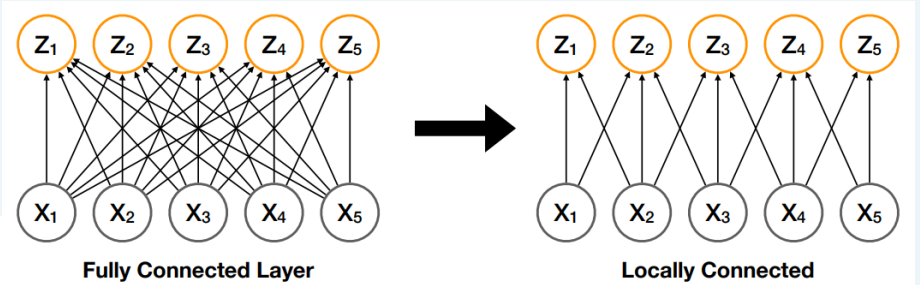
- **Cooperative** tasks
- Set up a **challenge** or **project**



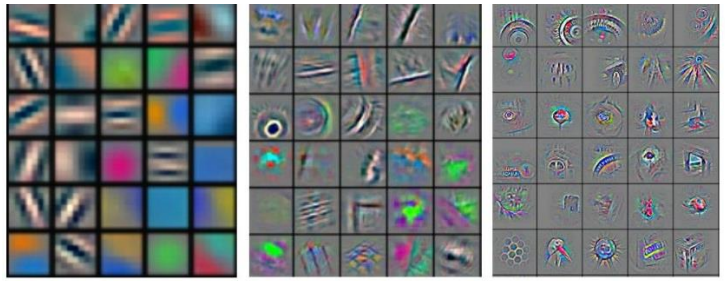
What might be hard to understand?

Images as input

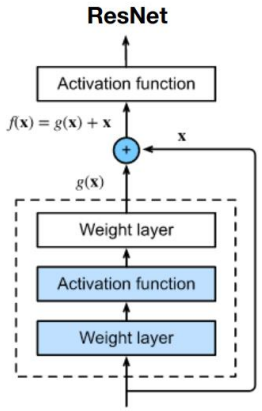
Concept of channels



Reasons for parameter reduction



What filters actually learn



Issues during the training process

Regularization & training tricks

How to set hyperparameters?