technische universität dortmund

fakultät für informatik

Bilderkennung mit Vgg

Proseminar „Convolutional Neural Networks - Methoden und Anwendungen“

Justin Schartner

6. Juni 2022

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Thema

Problemstellung

Entwicklung von genaueren CNNs

Lösung

Steigerung der Tiefe und minimale Filter

Ergebnis

Die Tiefe ist eine auschlaggebende Komponente

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

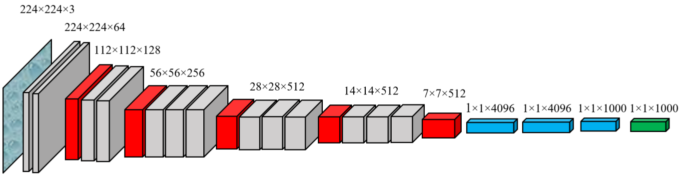
5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Idee



Steigerung der Genauigkeit durch:

- Steigerung der Tiefe, des CNNs
- Schachtelung von Convolutional-Layer-Blöcken
- Einsatz von kleinen 3x3-Filtern und einer Stride von 1

technische universität dortmund

fakultät für informatik

Aufgaben/ Einsatz

Bilderkennung

- Imagenet

  $\Rightarrow$ Klassifizierung in 1000 Kategorien

- Pneumoina Image

  $\Rightarrow$ Klassifizierung von Röntgenbildern

- Plant Image Classification

  $\Rightarrow$ Klassifizierung von Pflanzen

- ...

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

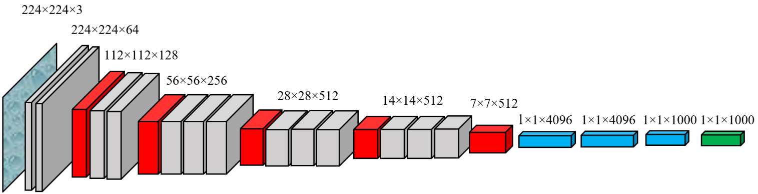
5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Architektur



convolution + ReLU

max pooling

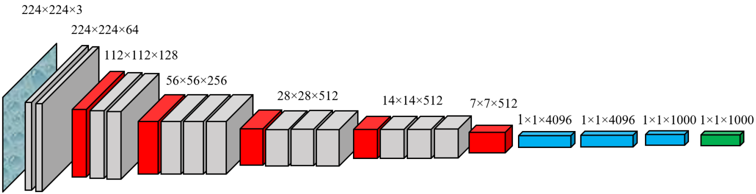
fully connected + ReLU

softmax

technische universität dortmund

fakultät für informatik

Architektur



convolution + ReLU

max pooling

fully connected + ReLU

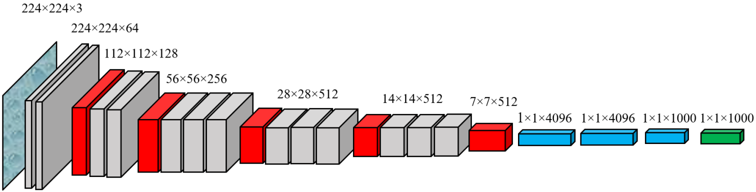
softmax

Merkmalsextraktion + klassisches MLP

Architekturarten

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 x 224 RGB-image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Architektur



convolution + ReLU

max pooling

fully connected + ReLU

softmax

Convolutional Layers

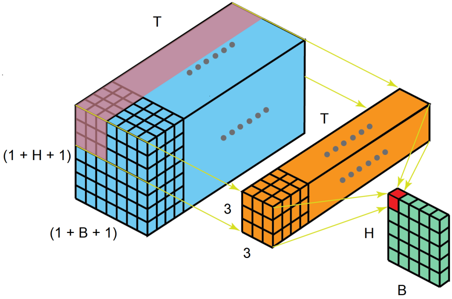
- 2D - Convolution
- Aktivierungsfunktion: ReLU
- Stride: 1x1
- Padding: 1, 0

  $\Rightarrow$ Die Breite und Höhe des Inputs wird beibehalten
- Kernel: 3x3 oder 1x1

  $\Rightarrow$ Minimaler Kernel für den Vergleich von Links/Rechts/Oben/Unten
- Filter Anzahl: 64, 128, 256, 512

  $\Rightarrow$ Filter lernen Muster des Inputs zu erkennen

Convolutional Layers

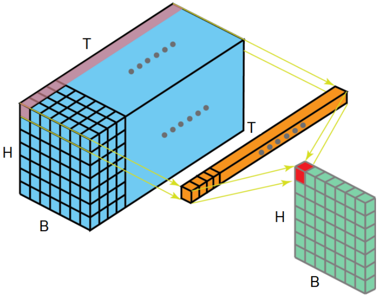


(Breite x Höhe x Tiefe) $\xrightarrow[\text{Conv2d(Filter: \textbf{3x3xTiefe}, Filter Anzahl: \textbf{n})}]{}$ (Breite x Höhe x n)

technische universität dortmund

fakultät für informatik

Convolutional Layers



(Breite x Höhe x Tiefe) $\xrightarrow[\text{Conv2d(Filter: \textbf{1x1xTiefe}, Filter Anzahl: \textbf{n})}]{}$ (Breite x Höhe x n)

technische universität dortmund

fakultät für informatik

ReLU
Rectified Linerar Unit

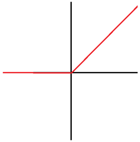


Abbildung: *ReLU*

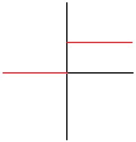
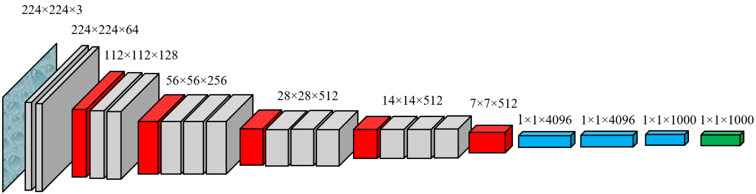


Abbildung: $ReLU'(x)$

$$ReLU(x) = \begin{cases} x & \text{x>0} \\ 0 & \text{sonst} \end{cases}$$

$$ReLU'(x) = \begin{cases} 1 & \text{x>0} \\ 0 & \text{x<0} \end{cases}$$

Architektur



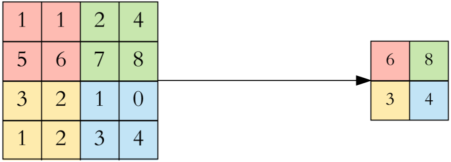
convolution + ReLU

max pooling

fully connected + ReLU

softmax

Max-Pooling

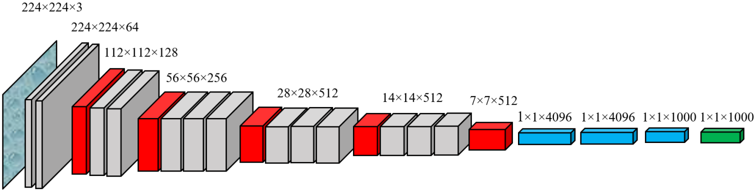


- Stride: 2x2
- Kernel: 2x2

  $\Rightarrow$ Die Breite und Höhe wird halbiert

  $\Rightarrow$ Daten werden auf die auschlaggebenden Informationen reduziert

Architektur



convolution + ReLU

max pooling

fully connected + ReLU

softmax

Fully Connected Layers

- Input: (7x7x512)

  ⇒ (7x7x512) Input-Neuronen

- Aktivierungsfunktion: ReLU
- Zwei versteckte Layer mit jeweils 4096 Neuronen
- ImageNet-Klassifiezunrg von 1000 Klassen

  ⇒ 1000 Output-Neuronen

  ⇒ Klassifiziert die gesammelten Information

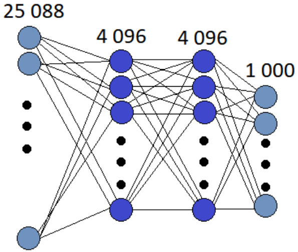
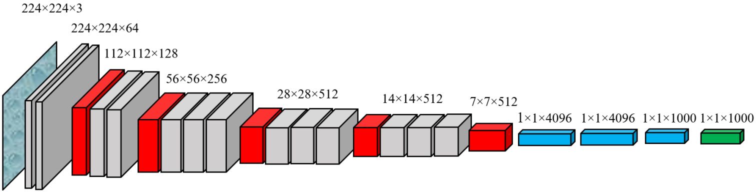


Abbildung: fully-connected layer

Architektur



convolution + ReLU

max pooling

fully connected + ReLU

softmax

technische universität dortmund

fakultät für informatik

Softmax

- normalisierte Exponentialfunktion
- kategoriale Verteilung
- Transformation in den Wertebereich [0,1]

$$\begin{pmatrix} -0.5 \\ 0.8 \\ 1.3 \end{pmatrix} \xRightarrow{Softmax} \begin{pmatrix} 0.093 \\ 0.342 \\ 0.564 \end{pmatrix}$$

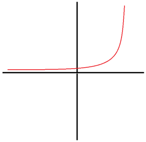


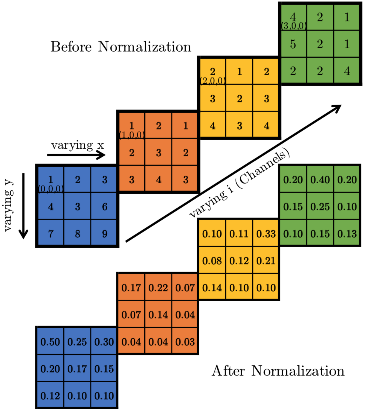
Abbildung: e^x

Softmax: $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

technische universität dortmund

fakultät für informatik

Local Response Normalization



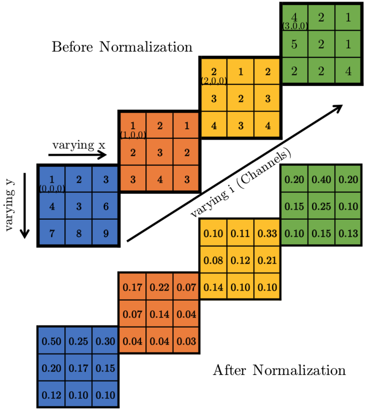
- Normalisation von Feature-Maps
- Nicht trainierbarer Layer

  $\Rightarrow$ Genauigkeit hat sich im VGG-net nicht verbessert

technische universität dortmund

fakultät für informatik

Local Response Normalization (Inter-Channel LRN)



$$N = 4$$

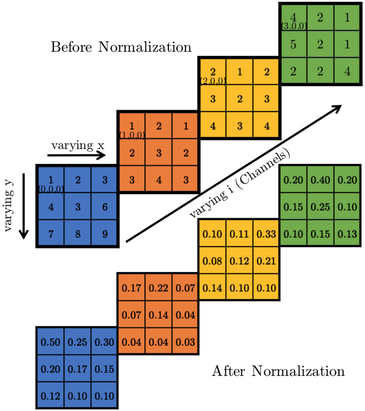
$$(k, \alpha, \beta, n)$$

$$b^i_{x,y} = a^i_{x,y} \cdot \frac{1}{\left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2\right)^\beta}$$

technische universität dortmund

fakultät für informatik

Local Response Normalization



$N = 4$

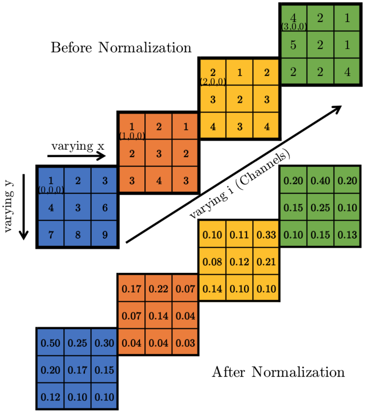
$(0, 1, 1, 2)$

$$b^2_{0,0} = a^2_{0,0} \cdot \frac{1}{\left(0 + 1 \sum_{j=1}^{3} (a^j_{0,0})^2\right)^1}$$

technische universität dortmund

fakultät für informatik

Local Response Normalization

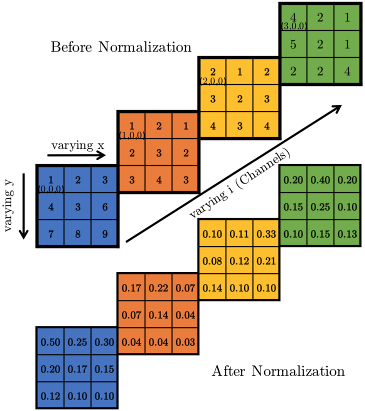


$N = 4$

$(0, 1, 1, 2)$

$$b^2_{0,0} = 2 \cdot \frac{1}{1^2 + 2^2 + 4^2}$$

Local Response Normalization



$N = 4$

$(0, 1, 1, 2)$

$b^2_{0,0} = 0.952$

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

technische universität dortmund

fakultät für informatik

Training

Optimierung des Traininigs durch:

- Batch-Size: 256
- **S**tochastic **G**radient **D**escent
  - $\Rightarrow$ Iteratives Verfahren zur Optimierung einer Funktion
  - $\Rightarrow$ Trainingszeit wird minimiert
- Dropout
  - $\Rightarrow$ Neuronen werden Netzwerk getrennt und wieder hinzugefügt
  - $\Rightarrow$ Beteilung von einzelnen Neuronen wird verringert
- L2-Regularization
  - $\Rightarrow$ Reduziert das Vorkommen von großen Gewichten
- Momentum
  - $\Rightarrow$ Ein zusätzlicher Faktor zur Learning-Rate
  - $\Rightarrow$ Lokale Minima werden gemieden

Bild-Augmentierung

Verfahren

- Skalieren
- Rotieren
- Ausschneiden
- Verschieben

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Vgg initialisieren

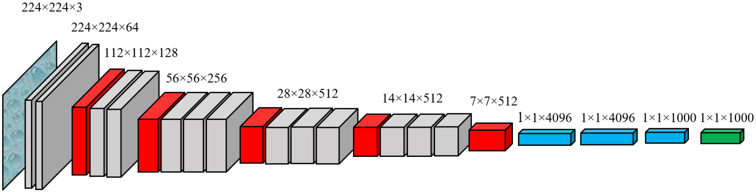
1. Variante: Die Komponenten erstellen und aneinander reihen.
   - Anpassbarkeit an das Problem
   - Das Netzwerk muss neu trainiert werden
2. Variante: Benutzen von schon bestehenden (und trainierten) Netzwerken.
   - Sie sind sofort einsatzbereit
   - Transfer Learning kann ausgenutzt werden
   - Das Netwerk bestimmt eine andere Klassifizierung

technische universität dortmund

fakultät für informatik

Vgg initialisieren

Architektur



convolution + ReLU

max pooling

fully connected + ReLU

softmax

Vgg initialisieren

1. Variante

```
def create_conv_layers(self, in_channels, architecture):
    layers = []

    #[[64, 64], [128, 128], [256, 256, 256], [512, 512, 512], [512, 512, 512]]
    for block in architecture:
        #[64, 64]
        for layer in block:
            out_channels = layer

            #conv-layer + relu
            layers += [
                nn.Conv2d(in_channels=in_channels, out_channels=out_channels,
                          kernel_size=(3,3), stride=(1,1), padding=(1,1)),
                nn.ReLU()
            ]

            in_channels = out_channels

        #max-pooling
        layers += [nn.MaxPool2d(kernel_size=(2,2), stride=(2,2))]
    return nn.Sequential(*layers)
```

Vgg initialisieren

1. Variante

```
    def create_fully_connected_layers(self, num_classes):
        fully_connected_layers = nn.Sequential(
            nn.Linear(512*7*7, 4096),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.Linear(4096, 4096),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.Linear(4096, num_classes),
            nn.Softmax(dim=0)
        )

        return fully_connected_layers
```

Vgg initialisieren

1. Variante

```
import torch.nn as nn

vgg16 = [[64, 64], [128, 128], [256, 256, 256], [512, 512, 512], [512, 512, 512]]
vgg19 = [[64, 64], [128, 128], [256, 256, 256, 256], [512, 512, 512, 512], [512, 512, 512, 512]]

class VGG_net(nn.Module):
    def __init__(self, in_channels=3, num_classes=1000, architecture=vgg16):
        super(VGG_net, self).__init__()

        #Convolutional Layers
        self.conv_layers = self.create_conv_layers(in_channels, architecture)

        #Fully Connected Layers
        self.fully_connected_layers = self.create_fully_connected_layers(num_classes)

    def forward(self, x):
        # x = [2, 3, 224, 224]

        x = self.conv_layers(x)
        # x = [2, 512, 7, 7]

        x = x.reshape(x.shape[0], -1)
        # x = [[1 .. (512x7x7)], [1 .. (512x7x7)]]

        x = self.fully_connected_layers(x)
        # x = [[1 .. 1000], [1 .. 1000]]

        return x
```

Vgg initialisieren

1. Variante

```python

if __name__ == "__main__":
    net = VGG_net(in_channels=3, num_classes=1000, architecture=vgg16)
    print(net)

```

technische universität dortmund

fakultät für informatik

Vgg initialisieren

1. Variante

```
VGG_net(
  (conv_layers): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU()
    (16): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU()
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU()
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU()
    (23): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU()
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU()
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU()
    (30): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
  )
  (fully_connected_layers): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
    (7): Softmax(dim=0)
  )
)
```

technische universität dortmund

fakultät für informatik

Vgg initialisieren

2. Variante

```python

import torch

if __name__ == "__main__":
    net = torch.hub.load('pytorch/vision:v0.10.0', 'vgg16', pretrained=True)
    print(net)


```

technische universität dortmund

fakultät für informatik

Vgg initialisieren

2. Variante

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
```

technische universität dortmund

fakultät für informatik

Vgg trainieren

```python
def train_network(net, x_train, y_train, epochs):

    #Training-Parameter
    learning_rate = 0.001
    momentum = 0.9

    #Loss function
    criterion = nn.CrossEntropyLoss()
    #Optimizer
    optimizer = optim.SGD(net.parameters(),
                          lr=learning_rate, momentum=momentum)

    for epoch in range(epochs):

        #reset the gradients in net
        optimizer.zero_grad()

        #forward
        outputs = net(x_train)

        #calculate loss
        loss = criterion(outputs, y_train)
        #calculate gradients in net
        loss.backward()

        #backward
        optimizer.step()
```

technische universität dortmund

fakultät für informatik

Vgg benutzen

```
#get network
net = torch.hub.load('pytorch/vision:v0.10.0', 'vgg16', pretrained=True)

#get categories
categories = loadCategories('imagenet_classes.txt')

#get image
image = Image.open('dog1.jpg')



#preprocess image
input_image = pre_process(image)

#forward
with torch.no_grad():
    output = net(input_image)



#apply softmax
probs = torch.nn.functional.softmax(output[0], dim=0)

#result
guess, guess_prob = findHighestProb(probs, categories)
print()
print(guess, guess_prob)
```

```
pug 0.9928255081176758
```

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Was ist aufgefallen

- Anzahl von Paramtern in einem VGG16: 134.7 Millionen (etwa 528MB)
  ⇒ Hoher Rechen- und Speicheraufwand

- Die Anwendung von mehreren 3x3 Filtern ersetzt die Funktionalität von bsp. 7x7 Filtern und erhört die Diskrimienitivität
- Die Tiefe des Netzwerkes hat die Komplexität erhöt

Ergebnis

Positiv

- Simples Design

  $\Rightarrow$ Leicht zu verstehen und zu implementieren

- VGG-Netzwerke sind erfolgreich, erzielen Resultate
- Design-Architektur hat auch andere CNN-Designs inspiriert

Negativ

- Speicherverbrauch
- Rechenaufwand

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

Ausblick

- Vgg-Netze werden mit anderen Neuronalen Netzen kombiniert

  $\Rightarrow$ Transfer Learning (kleiner Teil des Netzes wird neu trainiert)

- Convolutional Blöcke findet man in anderen Architekturen wieder
- Anwendung von 3x3-Filtern ist populärer geworden

technische universität dortmund

fakultät für informatik

Struktur

1 Einleitung

2 Allgemeines

3 Architektur

4 Training

5 Beispiel

6 Bewertung

7 Ausblick

8 Quellen

technische universität dortmund

fakultät für informatik

Quellen

Karen Simonyan & Andrew Zisserman | VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION | Visual Geometry Group | 2022 | 19.05.2022 | (https://arxiv.org/pdf/1409.1556.pdf)

Kunlun Bai | A Comprehensive Introduction to Different Types of Convolutions in Deep Learning | Towards Data Science | 2022 | 18.05.2022 | (https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215)

Aqeel Anwar | Difference between Local Response Normalization and Batch Normalization | Towards Data Science | 2022 | 06.06.2022 | (https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac)

Gaudenz Boesch | VGG Very Deep Convolutional Networks (VGGNet) - What you need to know | viso.ai | 2022 | 19.05.2022 | (https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/)

| Wikipedia | 2022 | 30.05.2022 | (https://en.wikipedia.org/wiki/File:VGG_neural_network.png)

Arden Dertat | Applied Deep Learning - Part 4: Convolutional Neural Networks | Towards Data Science | 2022 | 30.05.2022 | (https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2)

Zhi-Peng Jiang, Yi-Yang Liu, Zhen-En Shao & Ko-Wei Huang | An Improved VGG16 Model for Pneumonia Image Classification | MDPI | 2022 | 19.05.2022 | (https://mdpi-res.com/d_attachment/applsci/applsci-11-11185/article_deploy/applsci-11-11185.pdf?version=1637837328)

| ImageNet | 2022 | 19.05.2022 | (https://www.image-net.org/challenges/LSVRC/2014/results.php)

Mohamad Aqib Haqmi Abas, Nurlaila Ismail, Ahmad Ihsan Mohd Yassin & Mohd Nasir Taib | International Journey of Engineering & Technology | 2022 | 19.05.2022 | (https://www.sciencepubco.com/index.php/ijet/article/download/20781/9757)

Shreya Sinha & Duc V. Le | Completely Automated CNN Architecture Design Based on VGG Blocks for Fingerprinting Localisation | IEEE | 2022 | 31.05.2022 | (https://ieeexplore.ieee.org/abstract/document/9662642/authors#authors)