MASTER'S THESIS

Handpose-Estimation-Based Learning Academy for Improving Typing Efficiency

Mathias Mattersberger
m.mattersberger@student.tugraz.at

Supervisors: Priv.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner
Dipl.-Ing. Dr.techn. Josef Wachtler

Institute of Interactive Systems and Data Science

11. April 2024

TU Graz

Introduction

Deep Learning

Most frequently used terms in the context of current technological progress

Applications:

- Handpose estimation
- Object detection

Introduction
Touch Typing

Assign a specific key on the keyboard to each finger

Usecase:

- Increase typing efficiency

- More efficient use of computer tasks

- Healthier body/ finger posture leads to better ergonomics and less finger tiredness

Touch typing. url:https://de.wikipedia.org/w/index.php?title=Zehnfingersystem&oldid=236591813#/media/Datei:QWERTZ-10Finger-Layout.svg(visited on 04/02/2024)

Introduction

Combination of deep learning and touch typing

Idea to control touch typing with deep learning technologies

Realisation:

- Transfer keyboard scene stream using QR code technology
- Detect individual keys in the stream --> Object detection
- Detect visible hands and finger movements in the stream --> Hand pose estimation

Further explanation follows

Introduction

Typing Learning Academy

Structure:

- Based on the previously mentioned deep learning technologies

- Direct feedback on hand positions during typing

Introduction

Evaluation of the Typing Learning Academy

Study evaluation:

- Evaluate user experience and usability
- Evaluate the effectiveness of learning progress

Further explanation follows

Introduction

Impact & Motivation

First-time implementation of real-time **finger position detection** software for **touch typing**

Motivation:

- Apply deep learning technologies to verify correct finger position
- Improve touch typing learning progress with a web-based learning application

Overview Backround

A brief overview of important background knowledge

Convolutional Neural Networks (CNNs)
Image recognition, Different layers

Object Detection
Classification, Localisation, Applications

Hand Pose Estimation
Model structure

Convolutional Neural Networks (CNNs)

Image recognition

- Specific artificial neural networks

- Specially designed for grid data and analysing features in images

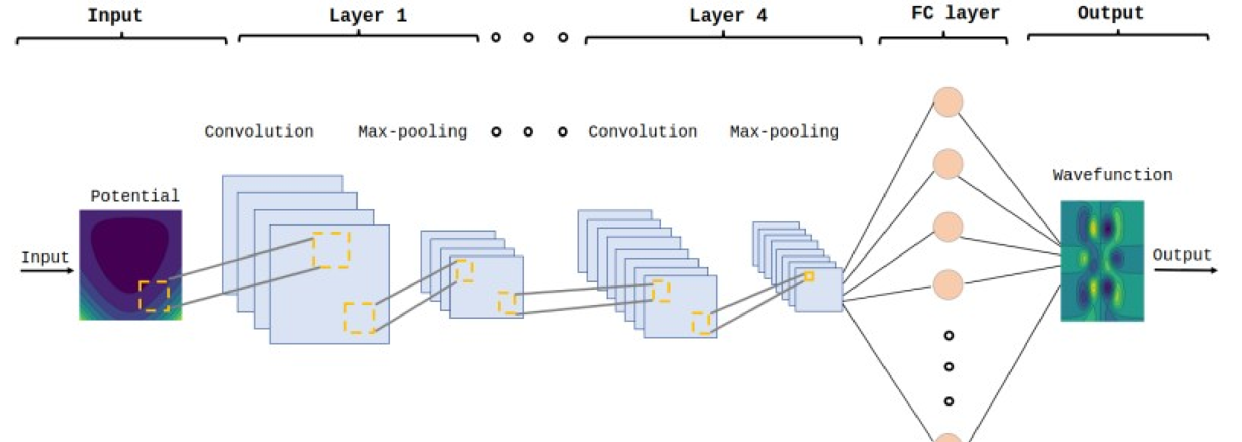
- Used in the field of image and video analysis

Convolutional Neural Networks (CNNs)

Different layers

- Input: Images as pixel matrices

- Convolutional layer
- Pooling layer
- Fully connected layer
- Output layer



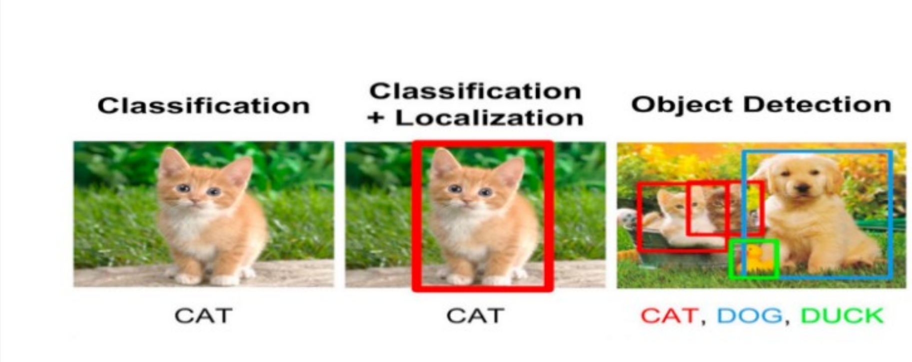
Design of a Convolutional Neural Network (CNN) [1]

Object Detection
Classification, Localisation

- Uses CNNs to find objects in the image

- Classification:
  - Get class of objects

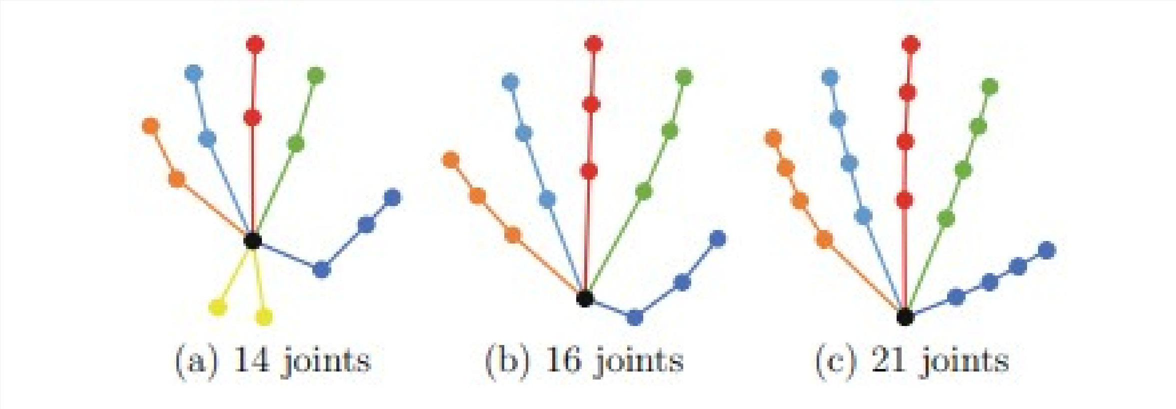
- Localisation:
  - Get position with bounding boxes



Difference of classification, localisation [2]

Hand Pose Estimation
Model structure

- Using large RGB image datasets to train the model
- Estimates hand postures and specific hand points
- Number of points may vary on trained model



Visualization of modeling hand with 14, 16 and 21 joints [3]

Overview Implementation

A brief overview ..

Video Stream Transmission
QR-Code technology, Socket.io, SimplePeer

Object Detection For Key Positions
General, Dataset, Tensorflow object detection API, EfficientDet architecture, Model training, Model validation

Hand Pose Estimation For Finger Positions
General, Mediapipe dataset, Palm detection model, Hand landmark model

Typing Learning Academy
General, Algorithm to check touch typing, Features

Video Stream Transmission

QR-Code technology

- Simple connection setup to stream keyboard scene

- QR code stream transmission should be fully automated

- Connection setup process with Socket.io and SimplePeer

Video Stream Transmission

Socket.io

- Protocol for bi-directional connections between client and server

- Each client is assigned to a socket instance

- Connection of two sockets (initiator/ client) handled by a unique ID

Video Stream Transmission

SimplePeer

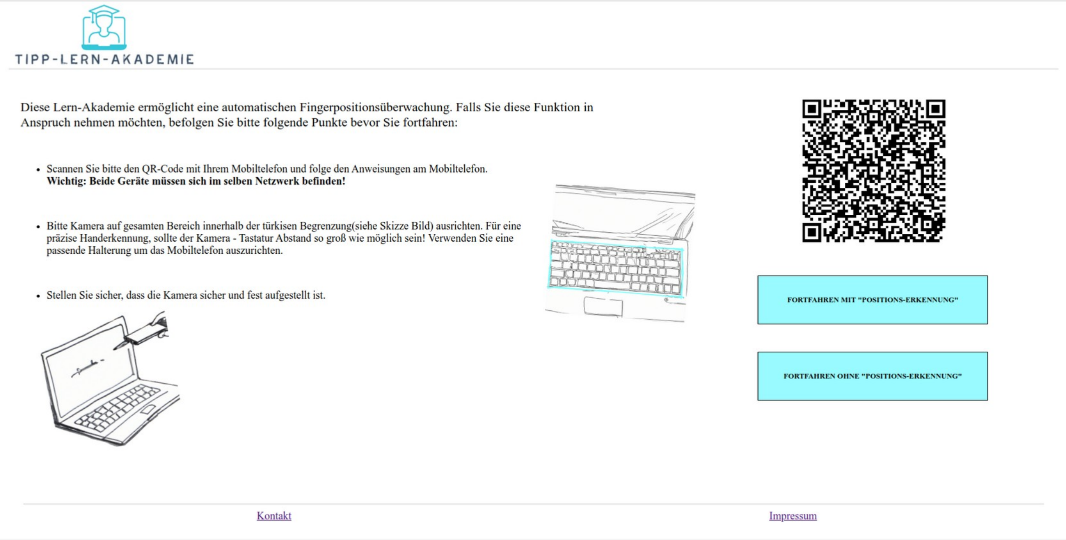
- WebRTC library for video exchange between peers

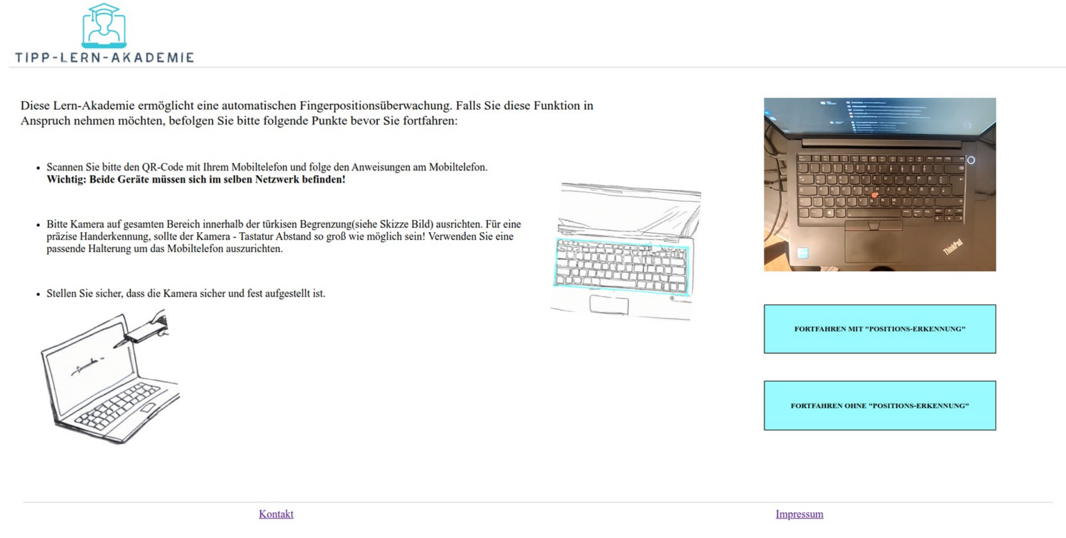
- Peer == End node (browser, mobile device)

- Fast and efficient data transfer

- Missing signalling server (socket.io)

Video Stream Transmission





Object Detection For Key Positions

General

- Train a convolutional neural network with custom dataset

- CNN should classify and locate keys on the keyboard with the help of image features

Object Detection For Key Positions

Datatset

- Object positions defined by bounding boxes in the image

- Object classes defined by labels in the image

- Completely new dataset for recognizing keys on a keyboard is annotated

Object Detection For Key Positions

Datatset

Object Detection For Key Positions

Datatset

- 5254 annotated images (60 labels per image)
- Robust dataset with images of real situations
- Data augmentation techniques used
- Split into training and test datasets

Object Detection For Key Positions

Tensorflow object detection API

- 40 different detection model architectures

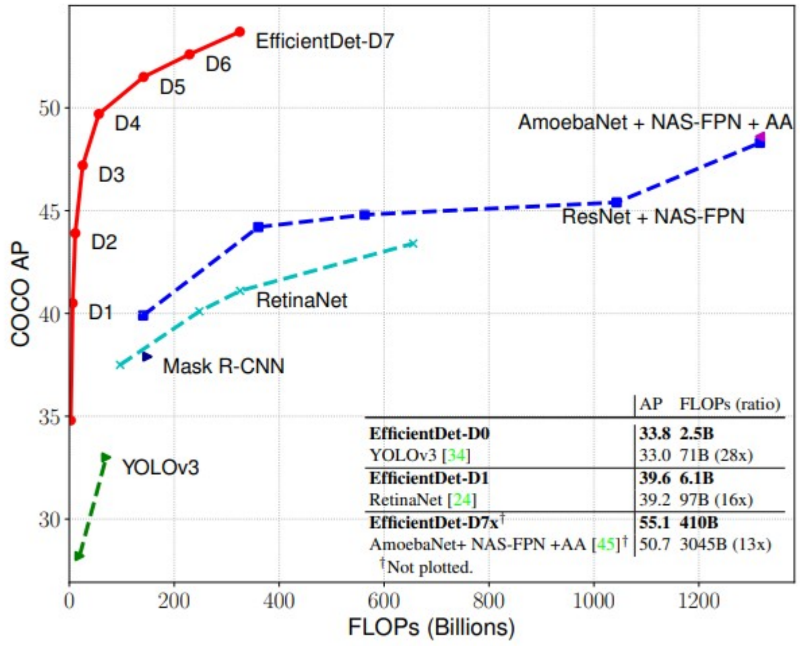
- Train and fine-tune custom object detection models based on provided architectures

Object Detection For Key Positions

EfficientDet architecture

- Fast inference time
- Acceptable mAP
- Small model size due to small number of model parameters and flops
- EffDet2: Good balance between speed and accuracy

Model FLOPs and related COCO accuracy of different architectures [4]



	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

Object Detection For Key Positions

Model training

- Based on the created **dataset** and the chosen **model architecture**

- Google Colab and Python

- Select appropriate **batch size** and **training steps** (overfitting)

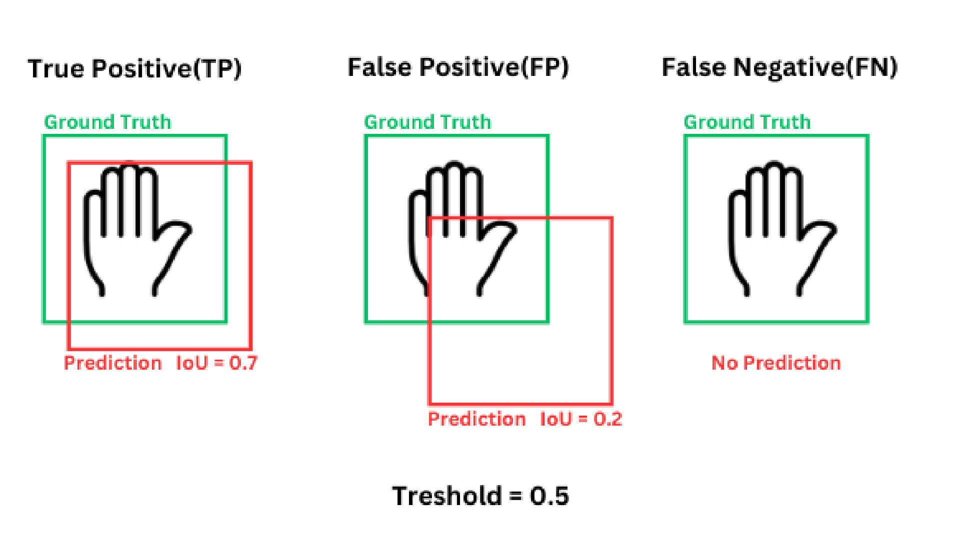
Object Detection For Key Positions

Model validation

- Based on the generated **test data** set

- Average precision

- Average recall



Object Detection For Key Positions

Model validation

EfficientDet D2 with 15000 Trainingsteps

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.848
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.966
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.951
```

```
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.877
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.882
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.882
```

Object Detection For Key Positions

Model validation

EfficientDet D2 with 7000 Trainingsteps

```
I1005 11:53:29.061347 134441826075776 model_lib_v2.py:1018]     + Loss/localization_loss: 0.022250
INFO:tensorflow:        + Loss/classification_loss: 0.148060
I1005 11:53:29.062944 134441826075776 model_lib_v2.py:1018]     + Loss/classification_loss: 0.148060
INFO:tensorflow:        + Loss/regularization_loss: 0.038689
I1005 11:53:29.064434 134441826075776 model_lib_v2.py:1018]     + Loss/regularization_loss: 0.038689
INFO:tensorflow:        + Loss/total_loss: 0.208999
I1005 11:53:29.066015 134441826075776 model_lib_v2.py:1018]     + Loss/total_loss: 0.208999
```

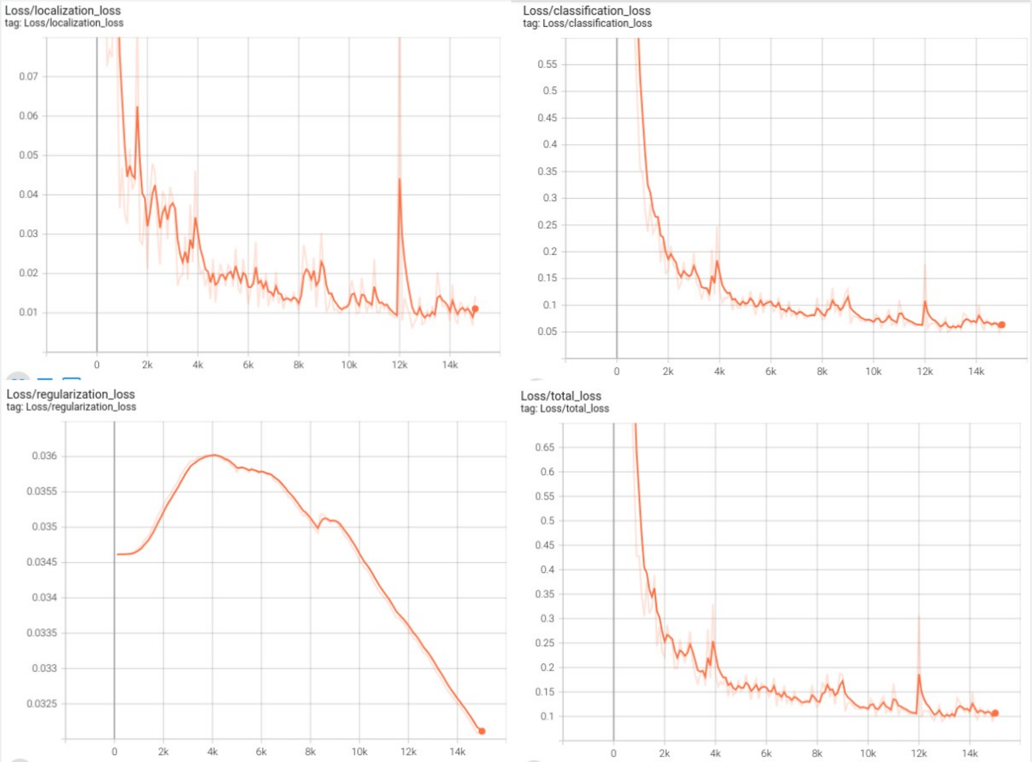
EfficientDet D2 with 15000 Trainingsteps

```
I0722 13:35:11.953575 136982381191808 model_lib_v2.py:1018]     + Loss/localization_loss: 0.013985
INFO:tensorflow:        + Loss/classification_loss: 0.102825
I0722 13:35:11.954401 136982381191808 model_lib_v2.py:1018]     + Loss/classification_loss: 0.102825
INFO:tensorflow:        + Loss/regularization_loss: 0.032064
I0722 13:35:11.955222 136982381191808 model_lib_v2.py:1018]     + Loss/regularization_loss: 0.032064
INFO:tensorflow:        + Loss/total_loss: 0.148874
I0722 13:35:11.956042 136982381191808 model_lib_v2.py:1018]     + Loss/total_loss: 0.148874
```

TU Graz

Object Detection For Key Positions

Model validation



Object Detection For Key Positions

Hand Pose Estimation For Finger Positions

General

- Real-time detection with two-stage hand tracking pipeline
    * Palm Detector
    * Hand Landmarker

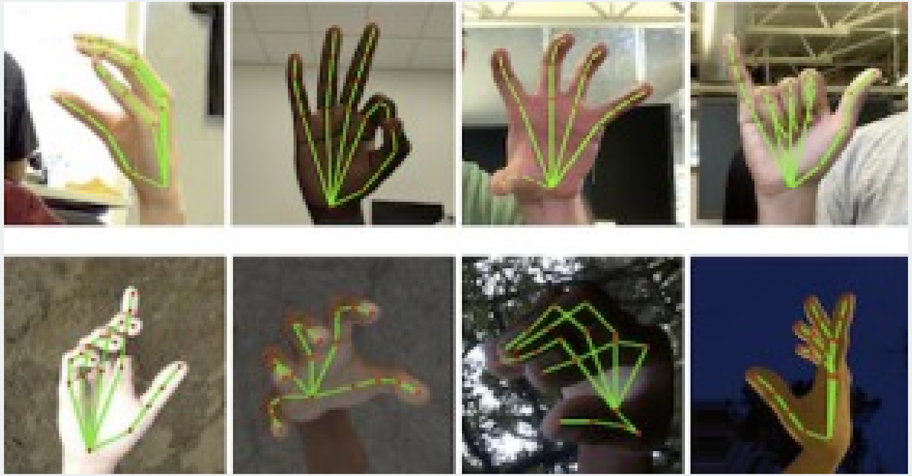
- Detect 21 hand landmarks from multiple hands

- MediaPipe Hand landmark model fits requirements

Hand Pose Estimation For Finger Positions

MediaPipe Dataset

- Provides real and synthetic image datasets of over 100.000 images



Dataset for MediaPipe handpose estimation [5]

Hand Pose Estimation For Finger Positions

Palm detection model

- Localise hands across the entire input image with bounding boxes

- Palms and fists are easier to detect (rigid object) than moving fingers

- Crop the area to the hand

Hand Pose Estimation For Finger Positions

Hand landmark model

- Applied to the previously performed palm detection

- Keypoint localisation of 21 hand points

- Detected hand points consist of x,y, and z coordinates

Hand Pose Estimation For Finger Positions

Typing Learning Academy

General

- Apply keyboard detection and hand pose estimation

- 170 learning courses with different levels of difficulty

Typing Learning Academy

Algorithm to check touch typing

- Key detection is applied once to save processing power
- Key bounding box positions stored in map
- Hand pose estimation is applied in real time
- Compare finger positions with detected key bounding box positions

Typing Learning Academy

Typing Learning Academy

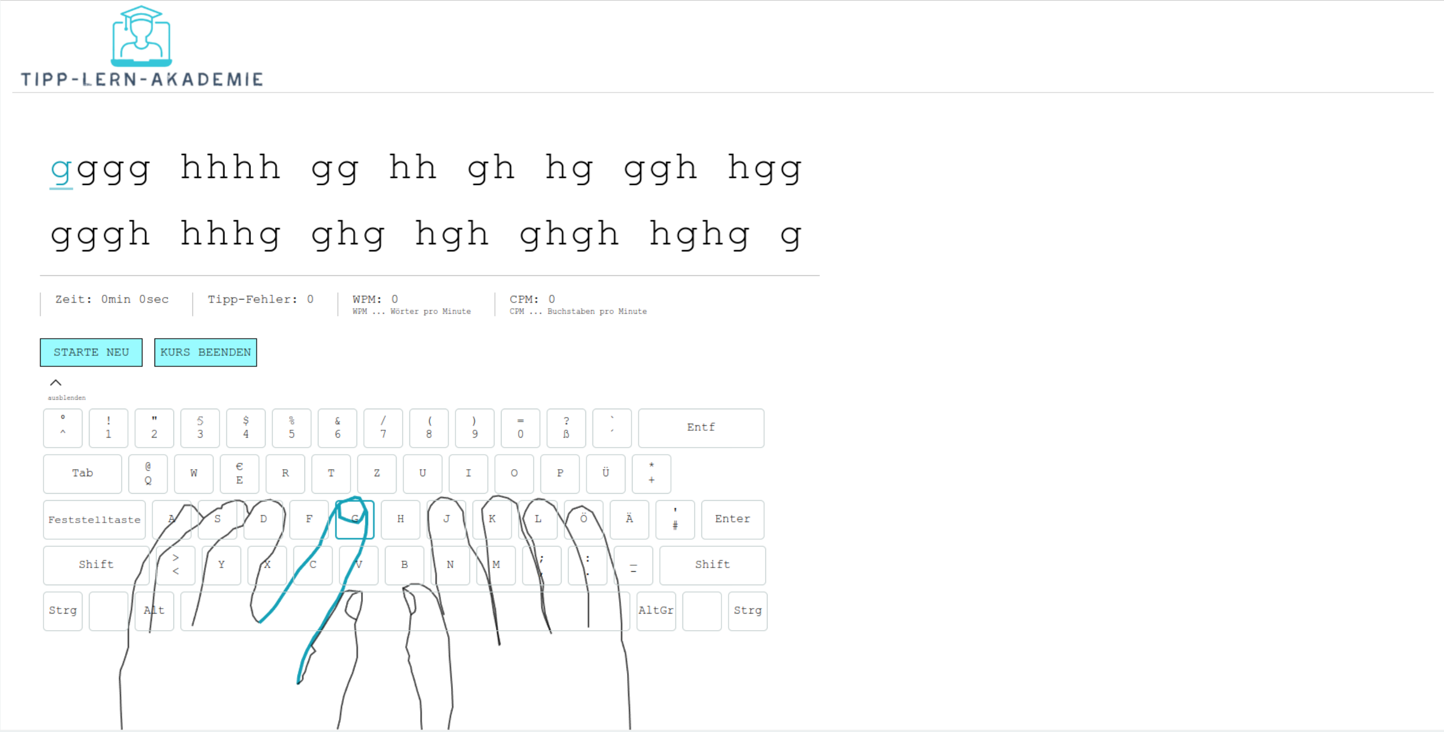
Features

- Virtual keyboard and hand sketch simulation

- Dynamic stream visualisation of key detection and hand pose estimation

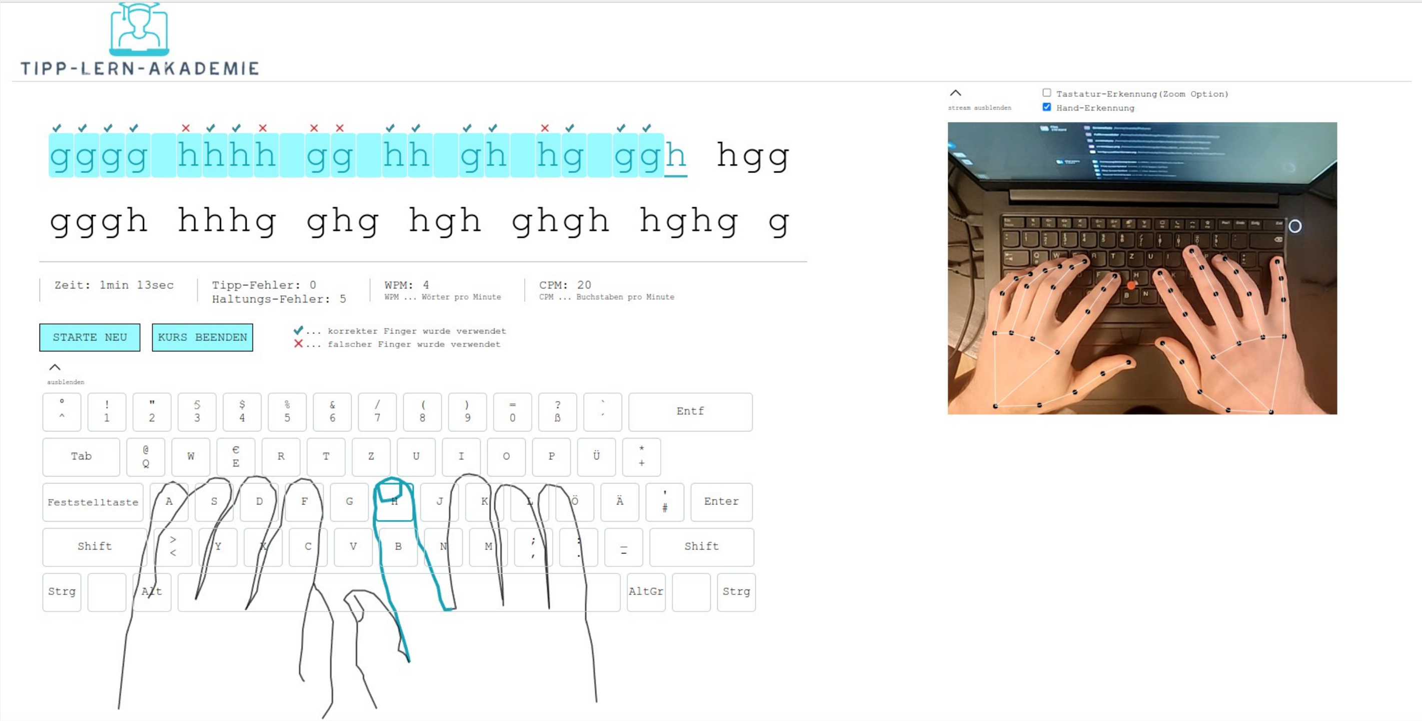
Typing Learning Academy

Feature: Virtual Keyboard and hand sketch



Typing Learning Academy

Feature: Direct position feedback



Overview Evaluation

A brief overview ..

Evaluation

General, Test persons, Study results

Evaluation
General

- Technical and didactic analysis
- Usability and technical errors of the Typing Learning Academy
- Learning progress of the Typing Learning Academy

Evaluation

Test persons

- 10 test persons selected for the study

- Potential main users of this learning web application

- Test group with certain diversity (technical understanding, experience in touch typing)

Evaluation

Study results

Positive Feedback:

- Usability, structure and different levels of difficulty

- Virtual keyboard and hand visualisation very helpful

- Test users paid particular attention to correct hand position

- Reduced error rate

Evaluation

Study results

Negative Feedback:

- Slow key recognition on laptops with low processing power

- Overlapping fingers lead to detection errors

- Fast typing lead to detection errors

- Test persons provide ideas for “future work”

Conclusion

- Approach to improve **touch typing** with **deep learning** technologies

- **Control** correct **finger position** in **real-time** browser environments

- **Improve** learning **progress** by using the Typing Learning Academy
    - More attention to finger posture
    - Correct finger position results in reduced typing error rate
    - Proven by the study participants

Bibliography

[1] L. Domingo and F. Borondo, “Deep learning methods for the computation of vibrational wavefunctions,” Communications in Nonlinear Science and Numerical Simulation, vol. 103, p. 105989, Dec. 2021. https://doi.org/10.1016/j.cnsns.2021.105989.

[2] Khan, Asharul (Apr. 2020), “Machine Learning in Computer Vision.” In: Procedia Computer Science 167, p. 1445. doi: 10.1016/j.procs.2020.03.355

[3] B. Doosti, “Hand Pose Estimation: A Survey,” arXiv.org, Jun. 02, 2019. http://arxiv.org/abs/1903.01013

[4] Tan, Mingxing, Ruoming Pang, and Quoc V. Le (2019). “EfficientDet: Scalable and Efficient Object Detection.” In: CoRR abs/1911.09070. arXiv: 1911.110Bibliography09070. url: http://arxiv.org/abs/1911.09070

[5] Zhang, Fan et al. (2020). “MediaPipe Hands: On-device Real-time Hand Tracking.” In: CoRR abs/2006.10214, pp. 1, 2. arXiv: 2006.10214. url: https://arxiv.org/pdf/2006.10214.pdf

TU Graz

Questions

Discussion

Thank you!

Mathias Mattersberger
m.mattersberger@student.tugraz.at

Institute of Interactive Systems and Data Science

Backup Slides

There is more…

Future Work

Multi-layer Artificial Neural Network

Convolutional Neural Network
Architecture, Convolution layer, Pooling layer, Fully connected layer

Object Detection
Dataset: Image Augmentation

EfficientDet Model
Architecture, EfficientNet backbone, BiFPN layer, Class/Box prediction network

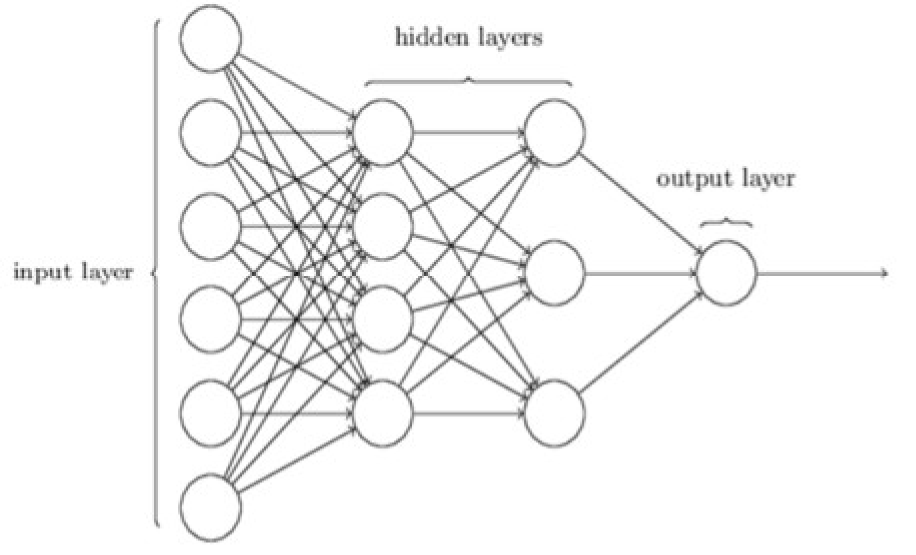
Future Work

- Extension of the data set
- Comparison of different model architectures
- Extension of the learning courses
- Conducting a long-term study
- Implement teacher and student accounts

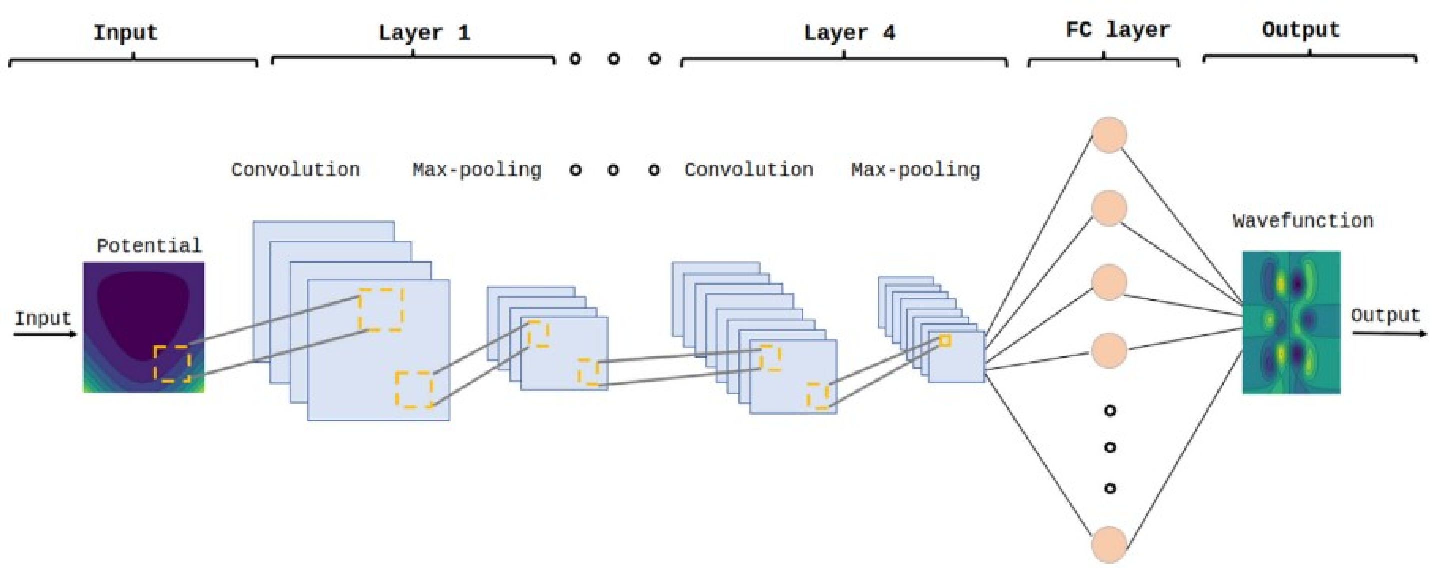
Multi-layer Artificial Neural Network

- Input layer
- Hidden layer
- Output layer

- Nodes are associated with weights
    - Weighted sum for each node



Convolution Neural Network
Architecture



Architecture of a convolutional neural network

Convolution Neural Network

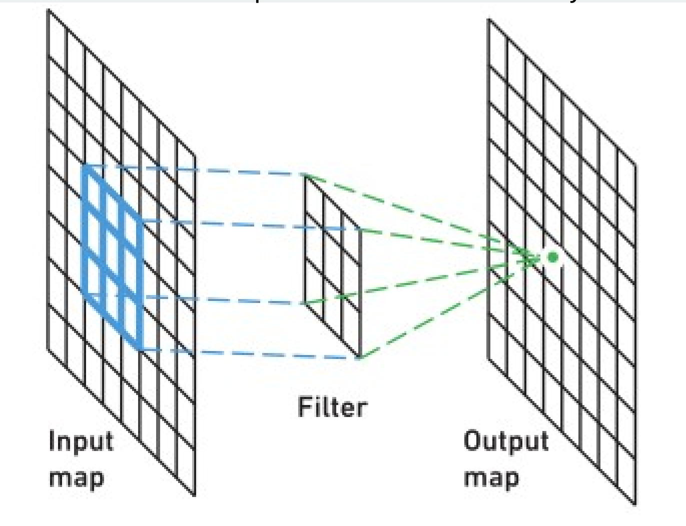
Convolution layer

- Extract features from image

- Multiply image matrix with filter kernel

- Different filters can be used (see next slide)

Output of the Convolutional Layer Filter



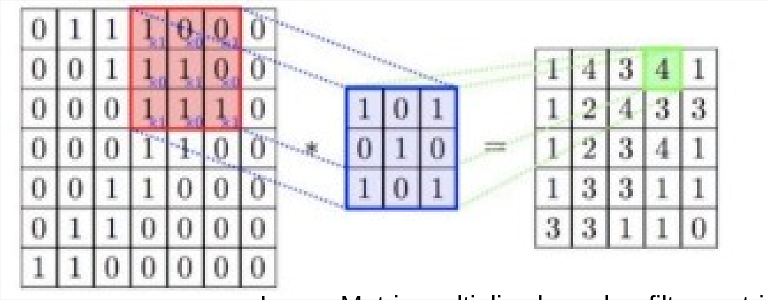


Image Matrix multiplies kernel or filter matrix

Convolution Neural Network

Convolution layer

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Examples of effects of different kernels on images

Convolution Neural Network
Pooling layer

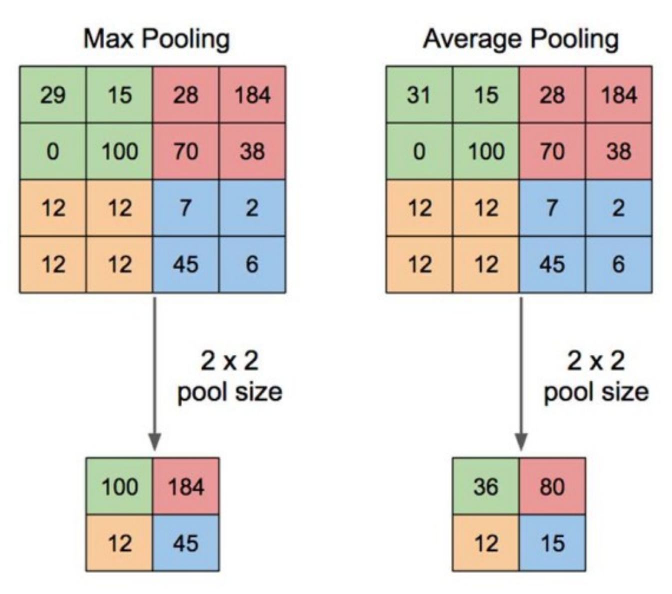
- Reduce dimension of the feature matrix (Convolution Layer)

- Reduce computational costs

- More complex structures are possible

- Max Pooling Average Pooling, Sum Pooling are applied to the feature matrix

Computation and Illustration of Max Pooling and Average Pooling



Convolution Neural Network

Fully connected layer

- Perform **classification process**

- Convert pre-processed image to **vector**

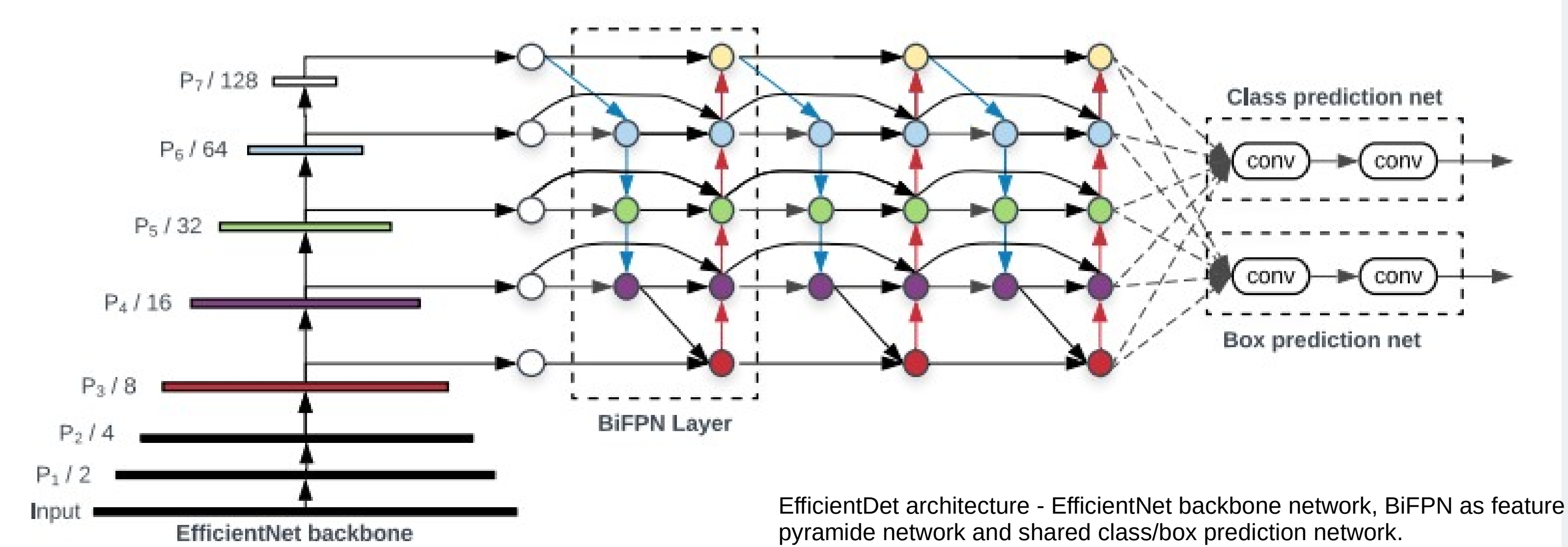
- Vector is input to trained neural network

- Neural network consists of **layers** and **neurons** with weighted connection

Object Detection

Dataset: Image Augmentation

EfficientDet Model
Architecture



EfficientDet architecture - EfficientNet backbone network, BiFPN as feature pyramide network and shared class/box prediction network.

EfficientDet Model

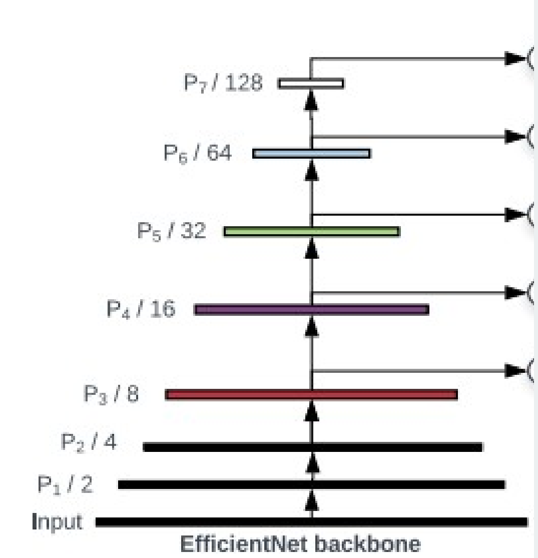
EfficientNet backbone

- Is a convolutional neural network

- Consists of seven layers (feature maps)

- Lower layers – finer details/ higher resolution

- Higher layers – abstract features/ lower resolution



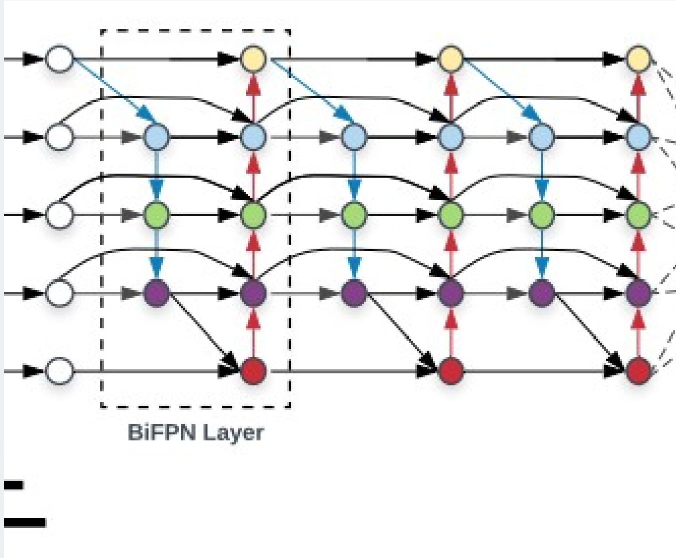
EfficientDet Model
BiFPN layer

- Solving scale inconsistency in CNNs

- CNNs have problems with objects of different sizes

- BiFPN layers are repeated for more accuracy

- Take feature levels (different dimensions) from EfficientNet Backbone and perform feature fusion from top to bottom and bottom to top



-Goal: Efficiently detect objects of different sizes

EfficientDet Model
Class/Box prediction network

- Predict object classes by **classification**
- Classification:
    - Assign bbox to a specific class and determine probability

- Predict bounding boxes by **regression**
- Regression:
    - Distribute anchor boxes across the image
    - Adjust anchor box position to match actual predicted bbox