

Goethe-Universität Frankfurt am Main

Institut für Informatik

Data Challenge-Numismatics



# Abschlussbericht

Vorgelegt von:

Leonidas Devetzidis

David Budgenhagen

Qudzia Farooq

Abgabedatum: 11.07.2021

# Inhaltsverzeichnis

<b>1. Challenge</b>	<b>2</b>
<b>2. Lösungsansatz</b>	<b>4</b>
<b>3. Analysen und Erkenntnisse</b>	<b>4</b>
3.1. Segmentierung der Münzen mittels naiven Computer Vision Methoden	4
3.2 Segmentierung der Münzen mittels anspruchsvollen Computer Vision Methoden	5
3.3 Objekterkennung der Münzen mit Deep Learning	6
3.3.1 Modell Auswahl	6
3.3.2 Trainingsdaten erstellen (annotieren)	7
3.3.3 Trainieren des Netzwerks	9
3.3.4 Evaluation	10
<b>4. Zukünftige weitere Analysen</b>	<b>12</b>

# 1. Challenge

Unsere Gruppe beschäftigt sich mit dem Ebay-Datensatz. Der Datensatz enthält Informationen zu vergangen Auktionen über versteigerte Münzen auf Ebay<sup>1</sup> in dem Zeitraum 03.09.2019 - 30.11.2019 und 01.12.2019 - 14.02.2020. Insgesamt ist der Datensatz 1.22 GB groß und beinhaltet Daten zu 2660 verschiedenen Münzauktionen. Die Daten sind im XML-Format gespeichert, wobei es zu jeder Auktion eine Beschreibung (im .txt Format) über die angebotene Münze und eine Vielzahl von Bildern gibt, die die Münze und ihren aktuellen Zustand beschreibt. Siehe Abbildung unten für die genaue Struktur einer Auktion der XML-Datei.

```
<?xml version="1.0" encoding="UTF-8"?>
- <findCompletedItemsResponse xmlns="http://www.ebay.com/marketplace/search/v1/services">
  <ack>Success</ack>
  <version>1.13.0</version>
  <timestamp>2019-12-01T17:52:46.939Z</timestamp>
  - <searchResult count="100">
    - <item>
      <itemId>372852101519</itemId>
      <title>Rom Kaiserreich - Unbestimmtes As mit Gegenstempel "B" </title>
      <globalId>EBAY-DE</globalId>
      + <primaryCategory>
        <galleryURL>https://thumbs4.ebaystatic.com/m/m2HRkiiXqWrTcGoVzq3TZFg/140.jpg</galleryURL>
        <viewItemURL>https://www.ebay.de/itm/Rom-Kaiserreich-Unbestimmtes-As-Gegenstempel-B-/372852101519</viewItemURL>
        <paymentMethod>CIPInCheckoutEnabled</paymentMethod>
        <paymentMethod>CashOnPickup</paymentMethod>
        <paymentMethod>PayPal</paymentMethod>
        <paymentMethod>PaymentSeeDescription</paymentMethod>
        <paymentMethod>MoneyXferAccepted</paymentMethod>
        <autoPay>false</autoPay>
        <postalCode>74***</postalCode>
        <location>Heilbronn,Deutschland</location>
        <country>DE</country>
      + <sellerInfo>
      + <shippingInfo>
      + <sellingStatus>
      + <listingInfo>
        <isMultiVariationListing>false</isMultiVariationListing>
        <topRatedListing>false</topRatedListing>
        <eBayPlusEnabled>false</eBayPlusEnabled>
      + <itemSpecifics>
    </item>
    + <item>
    + <item>
```

Bild 1.1: Die Struktur einer Auktion in der XML-Datei

Die folgenden Attribute sind in einer Auktion enthalten:

1. **itemId**: die einzigartige ID einer Auktion. Die ID ist dementsprechend wichtig, um die dazugehörige Beschreibung und die Bilder zuzuordnen.
2. **title**: der Titel der Auktion beinhaltet oft wichtige Details über die Münze.
3. **primaryCategory**: beinhaltet den Kategorienamen der Auktion.
4. **sellerInfo**: beinhaltet Informationen des Verkäufers, wie z.B. seinen Feedback-score oder seine positiven Bewertungen.
5. **sellingStatus**: beinhaltet Informationen über den Verkaufsstatus, wie z.B. der

---

<sup>1</sup> Für eine genaue Erklärung wie Ebay funktioniert, klicke [hier](#)

Verkaufspreis oder die anzahl Gebote.

6. **ItemSpecifics:** beinhaltet zusätzliche Informationen zu dem versteigerten Artikel, wie z.B. der Erhaltungsgrad oder das Material.

Die Abbildungen unten illustrieren ein Beispiel einer Auktion mit seinen Attributen und die dazugehörige Beschreibung und Bilder.

```
> 0006: <ebay_item.EbayItem object at 0x0000021DFA369040>
v 0007: <ebay_item.EbayItem object at 0x0000021DFA3694F0>
> special variables
  _bid_count: '14'
  _category_id: '18465'
  _category_name: 'Römische Kaiserzeit'
  _current_price: '25.5'
  _feedback_score: '36'
> _images: ['eBay-Daten\\Ebay_2019...5360_0.jpg', 'eBay-Daten\\Ebay_2019...5360_1.jpg', 'eBay-Daten\\Ebay_2019...5360_2.jpg']
  _item_description: 'Antoninian Gallienus Silber 3,15 g KPM 90.63 Aus alter Sammlung in Trier Echtheit Garantiert'
> _item_details: {'Motiv': 'Berühmte Persönlichkeit', 'Erhaltungsgrad': 'Sehr schön', 'Metall/Material': 'Silber'}
  _item_id: '174106805360'
  _positive_feedback_percent: '100.0'
  _title: 'Gallienus Antoninian Germanicus Maximius Top'
```

Bild 1.2: Die Attribute einer Auktion

Antoninian Gallienus Silber 3,15 g KPM 90.63 Aus alter Sammlung in Trier Echtheit Garantiert



Bild 1.3: Die Beschreibung der Auktion und die beigefügten Bilder

## 2. Lösungsansatz

Ziel ist es Münzen aus den gegebenen Bildern zu erkennen. Die Münzen sollten trotz Hintergrundrauschen, Okklusionen und anderen Objekten im Bild, zuverlässig erkannt werden. Für dieses Ziel kamen 3 Lösungsansätze zum Einsatz.

1. Segmentierung der Münzen mittels naiven Computer Vision Methoden
2. Segmentierung der Münzen mittels anspruchsvollen Computer Vision Methoden (Anthony, 2015)
3. Objekterkennung der Münzen mittels Deep Learning. Erkannte Objekte werden aus dem Originalbild zugeschnitten.

Die genaue Vorgehensweise beschreiben wir im nächsten Abschnitt.

## 3. Analysen und Erkenntnisse

### 3.1. Segmentierung der Münzen mittels naiven Computer Vision Methoden

Der naive Algorithmus zur Segmentierung der Münzen besteht aus mehreren Phasen:

1. Das Schwarz-Weiß-Bild wird mit einem Filter (z.B. Gaussian oder Median) gefaltet, um das Hintergrundrauschen zu reduzieren oder sogar zu entfernen.
2. Danach wird das transformierte Bild mit einem Sobel-Filter (Kanopoulos, 1988) gefaltet, um die Kanten des Bildes zu erkennen.
3. Im letzten Schritt wird der Watershed Algorithmus (Kanopoulos, 1988) angewendet, um die Münze vom Hintergrund zu segmentieren. Siehe Abbildung unten als Referenz.



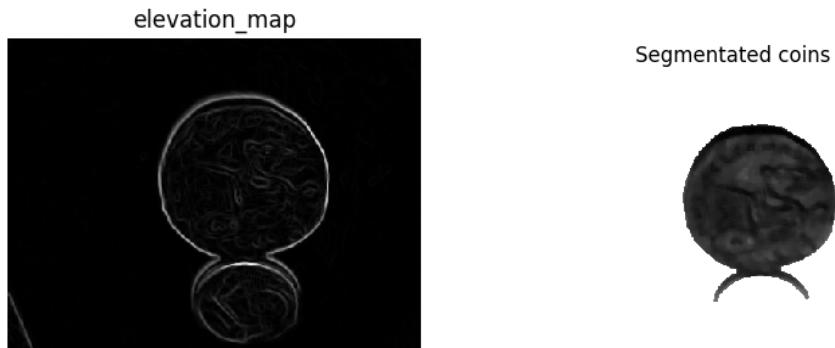


Bild 3.1: Zu sehen ist das Bild nach jedem Schritt des Algorithmus. Vom Originalbild (oben links) bis zum Ergebnis (unten rechts). Man erkennt am Ergebnis, dass der Algorithmus Münzen segmentieren kann, jedoch gibt es Probleme, wenn mehrere Münzen mit unterschiedlichen Farbtönen existieren.

### 3.2 Segmentierung der Münzen mittels anspruchsvollen Computer Vision Methoden

Der Algorithmus besteht aus den folgenden Schritten:

1. Das Bild wird in das HSV-Format<sup>2</sup> konvertiert um in Folge nur den Hue/Value Layer anzuzeigen.
2. Hue/Value Layer wird mit einem Median Filter gefaltet.
3. Adaptive threshold (Bradski and Kaehler, 2008) mit Gaussian wird angewendet.
4. Grab Cut (Carsten Rother, 2004) mit 5 Iterationen, um den Vordergrund vom Hintergrund zu trennen. Die Abbildung unten zeigt die Segmentierung der Münzen auf dem gleichen Bild wie im vorigen Algorithmus.

Segmented image



Bild 3.2: Das Ergebnis nach dem Grab Cut Algorithmus. Die Abbildung zeigt, dass der Algorithmus die Münzen segmentiert, jedoch gibt es einige Artefakte, die nicht entfernt werden.

---

<sup>2</sup> Für eine genauere Erklärung zum HSV-Format siehe [hier](#)

Nach den Anpassungen waren die Ergebnisse deutlich besser, jedoch gab es oft noch Artefakte (wie auf dem obigen Bild zu sehen), die nicht korrekt entfernt wurden. Außerdem hatte der Algorithmus Probleme, wenn das Bild viele Objekte (wie z.B. eine Waage o.ä.) enthält. Da der Algorithmus nicht auf dem ganzen Datensatz zuversichtlich die Münzen segmentieren konnte und wir unzufrieden mit den Ergebnissen waren, haben wir uns entschieden Deep Learning zu verwenden. Um genau zu sein, haben wir ein State of the art neuronales Netzwerk trainiert, dass Münzen erkennt und danach alle erkannten Münzen zuschneidet.

### 3.3 Objekterkennung der Münzen mit Deep Learning

Convolutional Neural Networks (CNNs) sind in der Lage Parallelverschiebungen eines Objekts zu erkennen und sind somit Translationsinvariant (Ian Goodfellow Et al., 2016). Dies ist eine wichtige Eigenschaft, da die Münzen auf den Bildern oft gedreht und/oder nicht immer im Zentrum des Bildes sind.

#### 3.3.1 Modell Auswahl

Hierzu kamen verschiedene State of the art Modelle in Frage. Wir entschieden uns für das EfficientDet Netzwerk (Mingxing Tan et. al., 2020), weil es sehr gute Ergebnisse bei einer vergleichsweise kompakten Architektur aufwies. Dies war wichtig, da uns kein Rechenzentrum zur Verfügung stand.

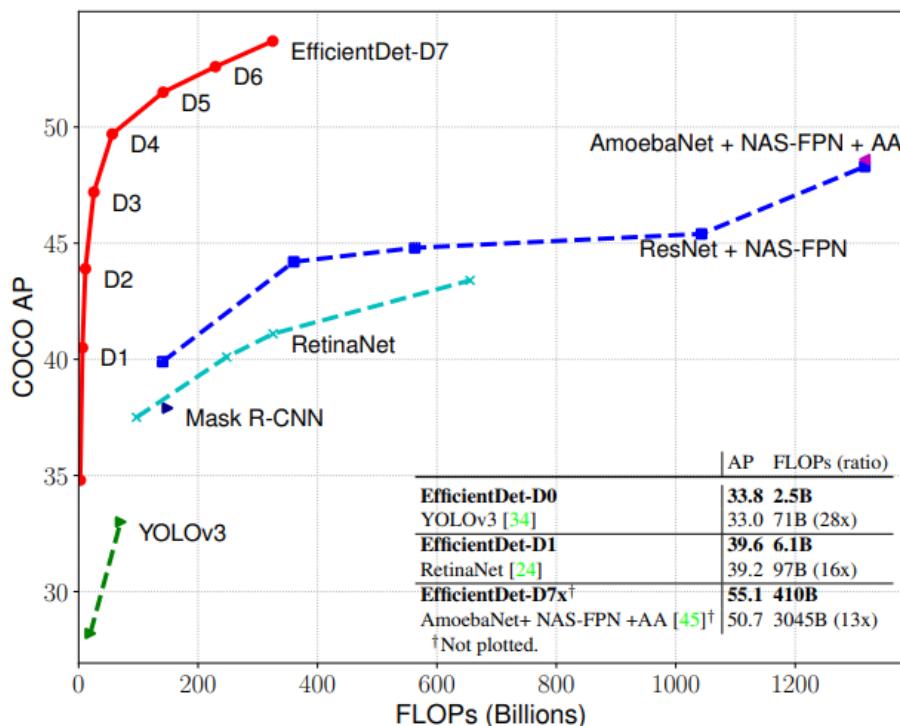


Bild 3.3: Model FLOPs vs. COCO accuracy (Mingxing Tan et. al., 2020).

Aus dem Bild wird deutlich, dass die EfficientDet Modelle D0-D7 vergleichbare oder sogar bessere performance auf dem COCO Datensatz, bei deutlich kompakterer Architektur (FLOPs) im Gegensatz zu anderen State of the art Modellen erzielen. Aufgrund unserer limitierten Ressourcen von 8GB Grafikspeicher wählen wir D0 und D1. Da wir schon bei D1 nur eine maximale batchsize von 4 wählen können ohne das wir einen “Out-of-memory error” erhalten.

### 3.3.2 Trainingsdaten erstellen (annotieren)

Um Trainingsdaten für das Netzwerk zu erzeugen, wurden 250 Bilder mit Hilfe des graphischen Bild-Annotierungstools LabelImg per Hand annotiert (tzutalin, 2017).

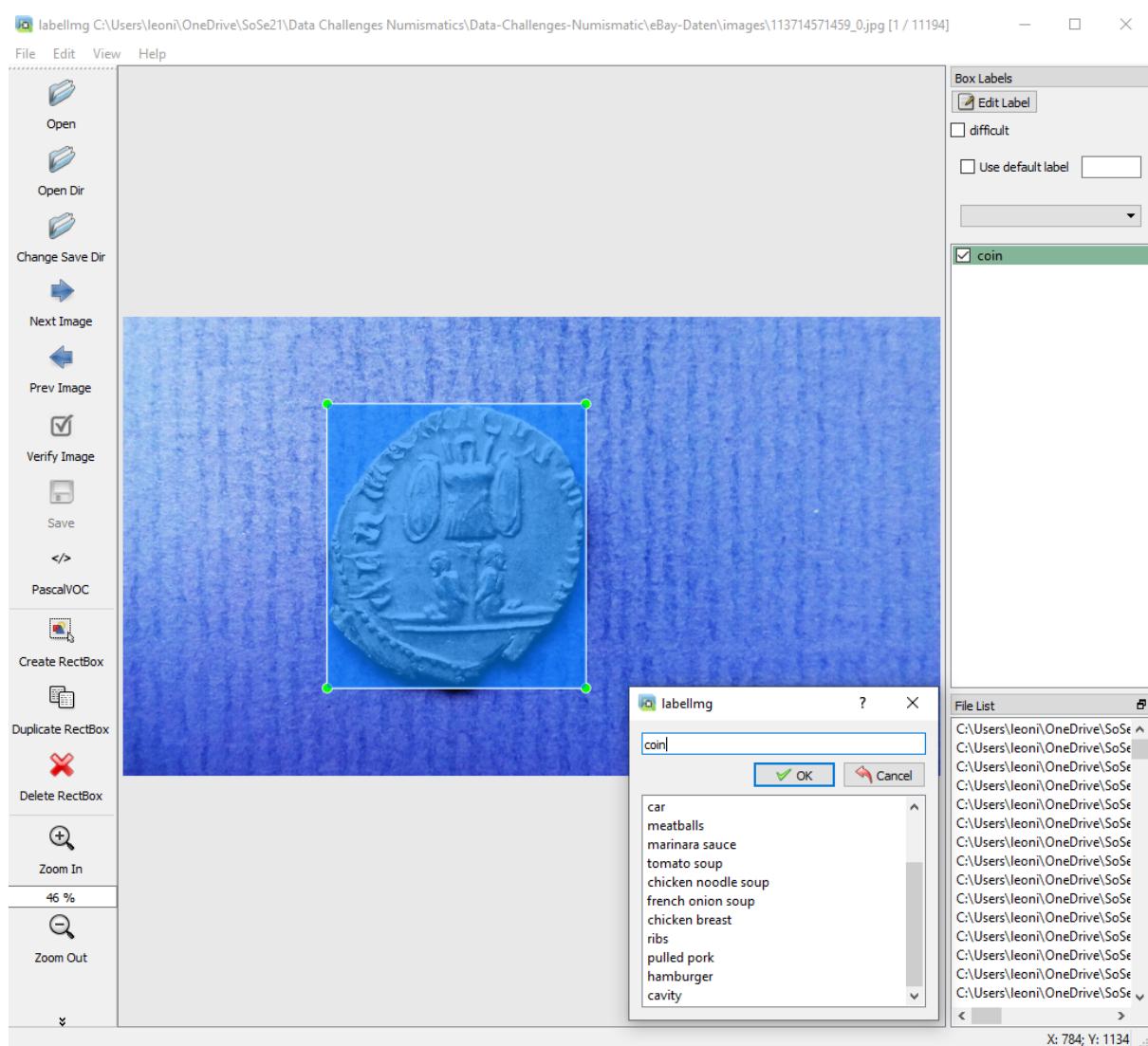


Bild 3.4: Beispiel einer manuellen Bild Annotierung mit LabelImg (tzutalin, 2017).

Mit einem Klick auf die Schaltfläche “Create RectBox” erstellt bzw. zieht man eine passende Bounding Box, um alle Münzen in einem Bild und trägt das gewünschte Label, in unserem

Fall "coin" ein. Als Resultat wird für jedes Bild eine .XML- Datei erstellt. Diese enthält die Koordinaten und Label für jede erstellte Bounding Box.

Als nächstes wird die Auflösung des Bild Datensatzes an die jeweiligen Netzwerke angepasst. EfficientDet D0 erwartet 512x512 Pixel große Bilder und EfficientDet D1 640x640 Pixel große Bilder (Mingxing Tan et. al., 2020).

### 3.3.3 Trainieren des Netzwerks

Um die vergleichsweise wenigen Trainingsdaten künstlich zu vervielfältigen und regularisierung zu betreiben nutzen wir Data Augmentation. Hier werden während der Trainingsphase modifizierte Kopien der Trainingsdaten erstellt. Dies erhöht die generalisierungsfähigkeit des Netzwerks (Ian Goodfellow Et al., 2016).



Bild 3.5: Data Augmentation: zeigt die möglichen modifizierte Kopien der Trainingsdaten.

Das erste Bild (links) zeigt das originale Bild. Das zweite Bild zeigt eine zufällig modifizierte rotation von 0-25°. Im dritten Bild wurde rotiert und vertikal gespiegelt. Im letzten Bild wurde rotiert, vertikal gespiegelt und etwas von den rändern abgeschnitten, indem eine zufällige Teilmenge des Originalbilds erstellt wird.

Beide Modelle wurden bereits vorgenommen auf dem COCO-2017<sup>3</sup> Datensatz. Modelle, die bereits vorgenommen sind, sind gut für die Erkennung von High-Level-Features wie Kanten, Muster usw. Diese Modelle verstehen bestimmte Feature-Repräsentationen, die wiederverwendet werden können. Das verschnellert den Trainingsprozess und sorgt für eine schnelle Konvergenz. Für das Training der Modelle EfficientDet D0/D1 wurde eine batch-size von 8/4 verwendet. Es wird empfohlen eine höhere batch-size für das Training zu verwenden, um eine schnelle Konvergenz zu gewährleisten. Der Grund für unsere Wahl ist, dass wir nur beschränkte Hardware zur Verfügung hatten (8GB Vram) und eine größere batch-size zu out-of-memory errors führten. Beide Modelle wurden für 43.000 Trainingsschritte trainiert. Die Trainings Metriken sind in der Abbildung unten zu sehen. Der classification\_loss ist wie gewöhnlich die cross-entropy loss function. Der localization\_loss ist der L1-loss zwischen den predicted bounding box corrections und den true values. Der regularization\_loss ist der Betrag, der durch die regularization function aggregiert wird. Die gesamten losses aufaddiert ergeben den total\_loss.

---

<sup>3</sup> Für weitere Details zum COCO-2017 Datensatz siehe [hier](#)

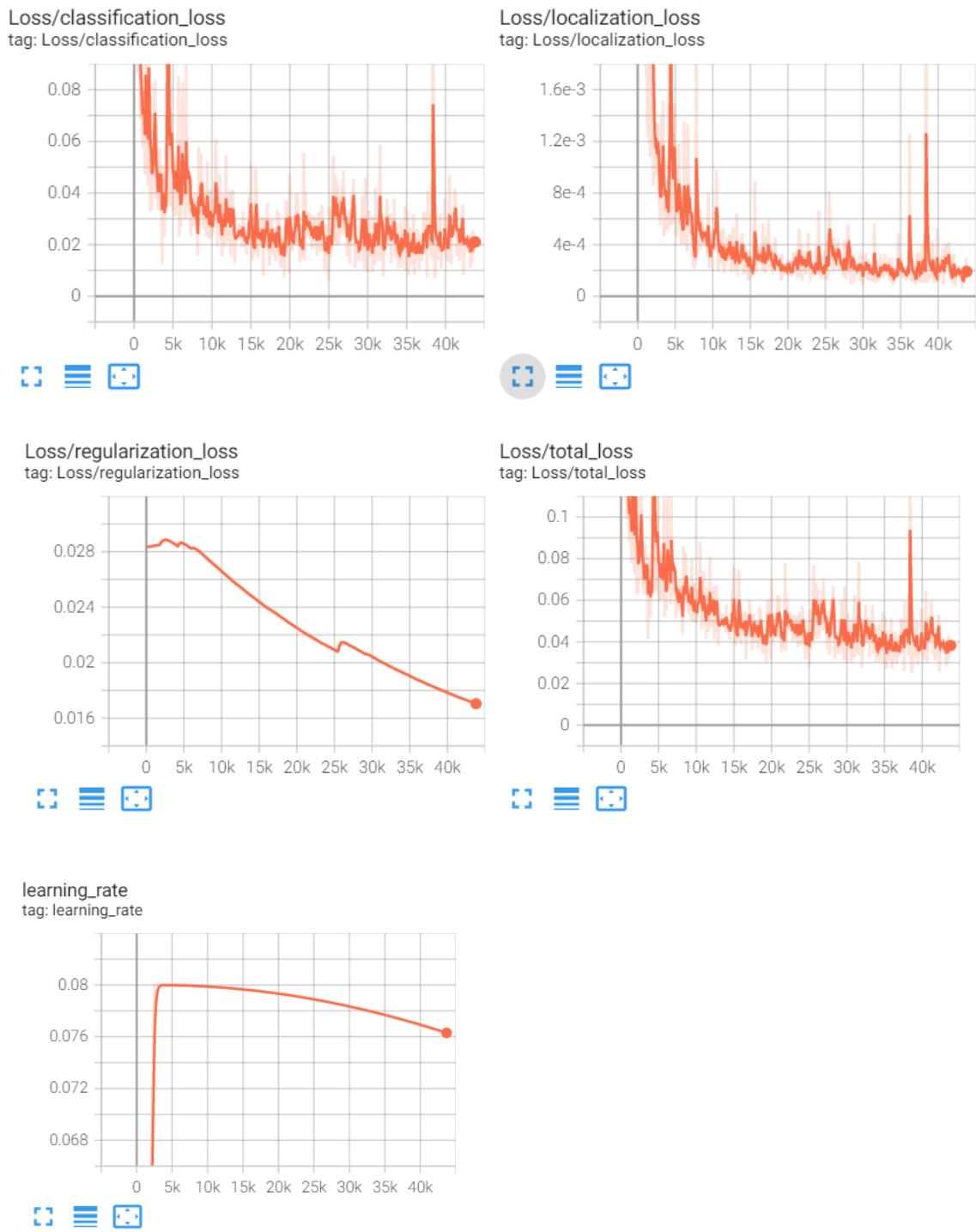


Bild 3.6: Die Metriken nach 43.000 Trainingsschritten

### 3.3.4 Evaluation

In computer vision ist mAP (mean average Precision)<sup>4</sup> die geläufigste Metrik in der Evaluation für die Objekterkennung (i.e. Lokalisierung und Klassifikation). Lokalisierung bestimmt den Standort einer Instanz (z. B. Bounding-Box-Koordinaten) und die Klassifizierung bestimmt die Instanz. Die Tabelle unten zeigt die Evaluation für das EfficientDet D1. Die Ergebnisse waren vergleichbar ähnlich zu dem EfficientDet D0, wobei das EfficientDet D1 ungefähr 1-2% besser abgeschnitten hat. Die Evaluation wurde auf 20% der Daten ausgeführt (insgesamt nur 50 Bilder). Da es aus Zeitgründen nicht möglich war mehr Bilder zu annotieren, wäre es für weitere Analysen interessant eine anspruchsvollere Architektur zu verwenden (z.B. EfficientDet D7) mit einem größeren annotierten Datensatz.

Metrik	Intersection of Union (IoU)	mAP
Average Precision	IoU=0.50:0.95	0.920
Average Recall	IoU=0.50:0.95	0.942

Da wir nur eine Klasse haben für die Klassifizierung (Coin), ist der mAP = AP = 0.92.

---

<sup>4</sup> Geläufige Metriken (u.a. mAP) für Objekterkennung sind [hier](#) im Detail erklärt

Siehe die Abbildungen unten für diverse Ergebnisse

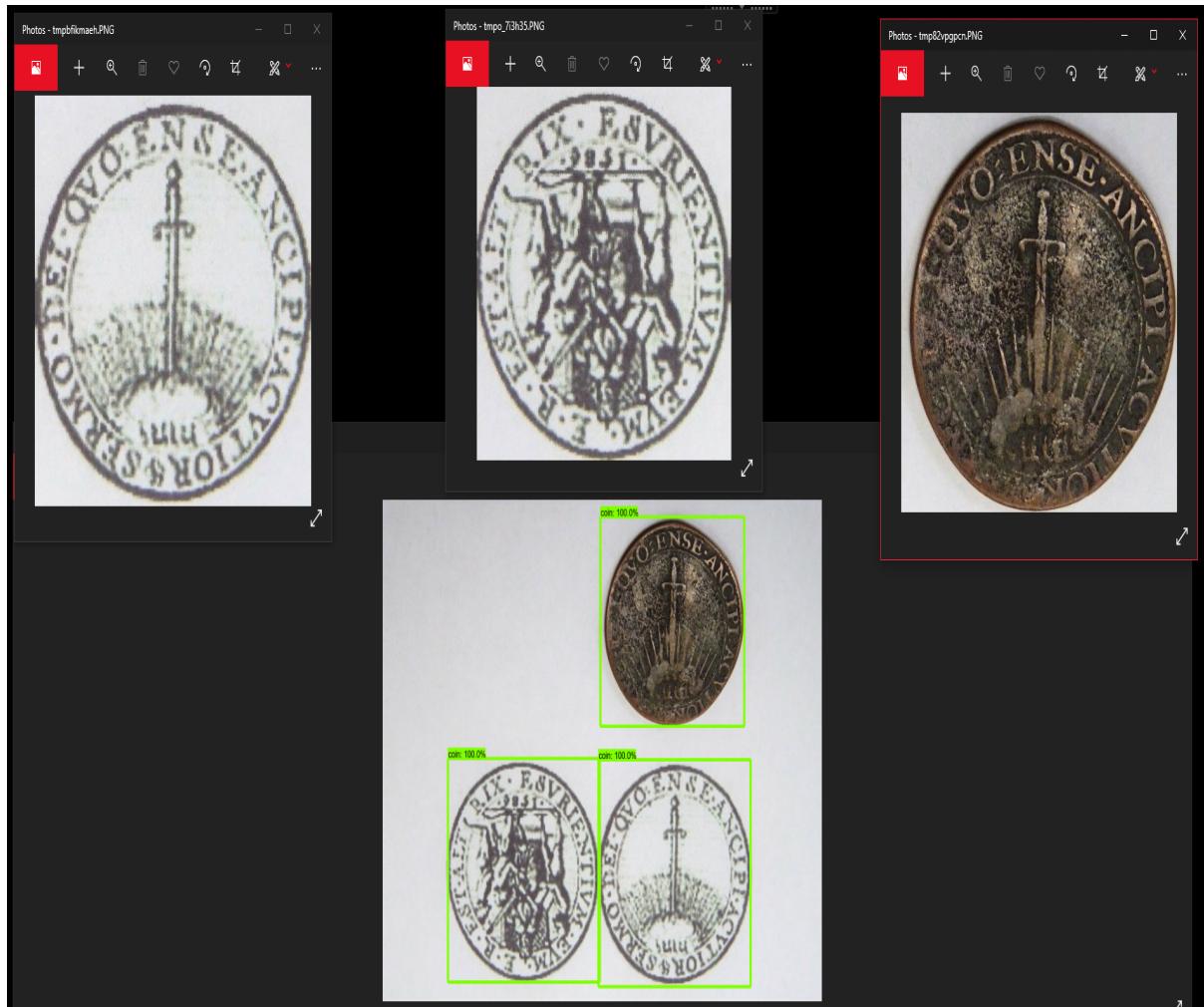


Bild 3.7: Das Originalbild (unten) und die erkannten Münzen (oben), die mit 100% Sicherheit erkannt wurden. Die erkannten Münzen werden danach aus dem originalen Bild zugeschnitten und separat gespeichert.



Bild 3.8: Mehrere Münzen werden gleichzeitig erkannt (links). Trotz starkem Hintergrundrauschen wird die Münze auf der Waage erkannt (rechts).

In der Praxis funktioniert die Deep Learning Variante für die Segmentierung verschiedener Instanzen (in unserem Fall Münzen) am Besten.

## 4. Zukünftige weitere Analysen

Aus Zeitgründen konnten einige Analysen nicht durchgeführt werden. Dieses Projekt dient als Basis für noch weitere interessante Analysen der Objekterkennung. Mithilfe der Objekterkennung könnten nicht nur Münzen, sondern auch antike- oder Euromünzen erkannt und unterschieden werden. Zusätzlich wäre es noch interessant die erkannten antiken Münzen auf weitere Merkmale wie Münzbild, Größe, Prägejahr, Gewicht, Material, Nominal Bezeichnung, Herkunft sowie Ausgabeland bzw. Kategorie zu klassifizieren.

Bei Verfügbarkeit von ausreichender Hardware (z.B. Uni Sever) könnten anspruchsvollere Modelle, wie z.B. „EfficientDet-D7“ mit einer größeren batch-size trainiert werden, was wiederum zu besseren Ergebnissen führen würde.

Interessant wäre auch mehr finetuning an den Modellen vorzunehmen. Die Auswahl der Hyperparameter (z.B. learning rate, loss functions etc.) der Modelle haben eine große Auswirkung auf die Konvergenz des Trainings und auf die Ergebnisse des Netzwerks.

Schließlich wäre es auch sinnvoll, das Modell mit viel mehr Daten zu trainieren als nur 250 Bildern. Das jetzige Modell hat vor allem Probleme, wenn es Münzen erkennen soll, die es vorher im Training nicht gesehen hat.

Der Datensatz umfasst 11194 Bilder. Die Annotierung und Verwendung aller Bilder für das Training, würde für ein viel robusteres Modell sorgen. Mehrere Trainingsdaten würden eine bessere Generalisierung gewährleisten.

## Bibliography

- Anthony. (2015). *Coin Detection*. <https://andrewbfang.com/resources/coin.pdf>
- Bradski and Kaehler. (2008). *Learning OpenCV*.
- Carsten Rother. (2004). "GrabCut": interactive foreground extraction using iterated graph cuts.
- Ian Goodfellow Et al. (2016). *Deep Learning*. MIT Press.
- Jos B. T. M. Roerdink Jos B. T. M. Roerdink View Profile. (2003). *The watershed transform: definitions, algorithms and parallelization strategies*.
- Kanopoulos. (1988). *Design of an image edge detection filter using the Sobel operator*.
- <https://ieeexplore.ieee.org/document/996>
- Mingxing Tan et. al. (2020). *EfficientDet: Scalable and Efficient Object Detection*.
- <https://arxiv.org/pdf/1911.09070.pdf>
- tzutalin, d. (2017). *LabelImg*. <https://github.com/tzutalin/labelImg.git>