

AI-Document-Checker

Alin Bicer

Berre Nur Celik

Egesel Bozgeyik

(Juli 2025)

0. Einleitung zum Dokument

Dieses Projekt ist das Abschlussprojekt des Moduls INF2207 Grundlagen der Data Science an der Technischen Hochschule Mittelhessen.

In der Dokumentation haben wir versucht, uns so genau wie möglich an den Text zu halten, der das Projekt beschreibt. Das Projekt besteht aus einem von uns trainierten Bild Erkennungsmodell, das mit dem qwen 2.5 Multimodal LLM auf einem von der THM zur Verfügung gestellten Computer zusammenarbeitet, um die vom Benutzer eingegebene Musterlösung und Abgabe zu vergleichen und zu bewerten.

1. Mögliche Modelle zur Verwendung

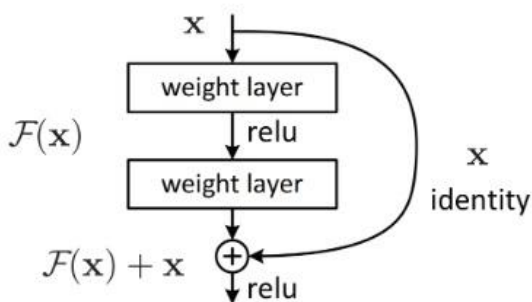
1.a. ResNet

Einleitung auf das Modell

Bis in die 2010er Jahre war Deep Learning noch nicht so weit entwickelt wie heute. Die Gründe dafür waren der Mangel an verfügbaren Daten und Hardware sowie die Schwierigkeit, Modelle zu trainieren. Im Jahr 2012 schufen die Entwickler AlexNet. AlexNet gewann den ImageNet-Wettbewerb mit einem Erdrutschsieg. Dieser 8-Schicht-CNN war damals revolutionär. Im Jahr 2014 wurden auch VGGNet und GoogLeNet eingeführt. Sie brachten das Deep Learning bei der Überwindung einiger Probleme wie der Überanpassung ein Stück weiter. Allerdings funktionierte die Erhöhung der Tiefe nicht immer, und in vielen Fällen schnitten die Modelle mit tieferen Schichten nicht besser ab. Im Jahr 2015 stellten 4 Forscher von Microsoft Research Asia ResNet vor. Dieses Modell, das für Residual Network steht, kann bis zu 152 Schichten umfassen und löste das Problem der Leistungsverschlechterung mit zunehmender Anzahl von Schichten in anderen Modellen. Mit diesem Modell gewannen die Forscher den 1. Platz im ImageNet-Wettbewerb 2015. Mit der Einführung dieses Modells wurden tiefere Modelle durch residuales Lernen möglich. In den folgenden Jahren bildete ResNet den Grundstein für viele neu entwickelte Modelle.

Im Gegensatz zu früheren Ansätzen geht es bei ResNet, wenn CNN zwischen den Schichten operiert, darum, wie sehr sich die Ausgabe von der Eingabe unterscheidet, um das Ergebnis der Funktionsoperation zu erreichen. Es lässt die Eingabe so, wie sie ist, und konzentriert sich nur auf das, was hinzugefügt werden muss. Mit diesem Ansatz wurde das Problem des Verschwindens des Gradienten gelöst, das bei früheren Modellen auftrat.

Theoretischer Hintergrund



Die Idee: Bei früheren Lernmethoden bestand das Ziel darin, den Output direkt zu berechnen. Bei ResNet hingegen geht es nicht um die Berechnung von $H(x)(\text{Output})$, sondern darum, zu berechnen, wie stark wir von x abweichen.

Die Ausgabe dieses Netzes:

Ausgabe: $F(x) + x$

Diese Struktur wird Residual Learning genannt, weil nicht der Output direkt gelernt wird, sondern die Differenz zwischen Output und Input.

Das Lernen erfolgt im Allgemeinen durch Aktualisierung der Gewichte zwischen den Verbindungen. Diese Aktualisierung erfolgt mit Hilfe des Gradienten. Mathematisch gesehen ist der Gradient die Ableitung der Verlustfunktion in Bezug auf das Gewicht.

Neues Gewicht = Gewicht - (Gradient x Lernrate)

Dieser Vorgang wird bei jedem Schritt wiederholt, und auf diese Weise beginnt das Netz, bei jedem Schritt ein wenig genauer vorherzusagen.

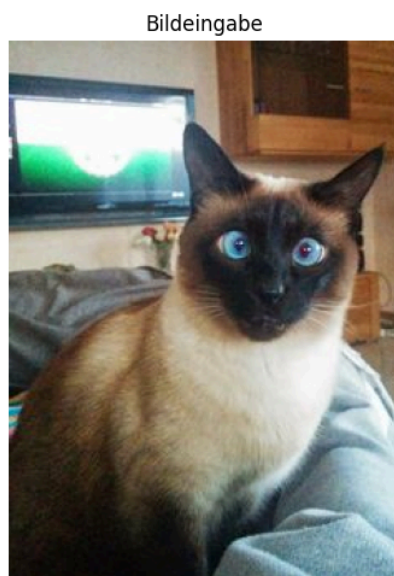
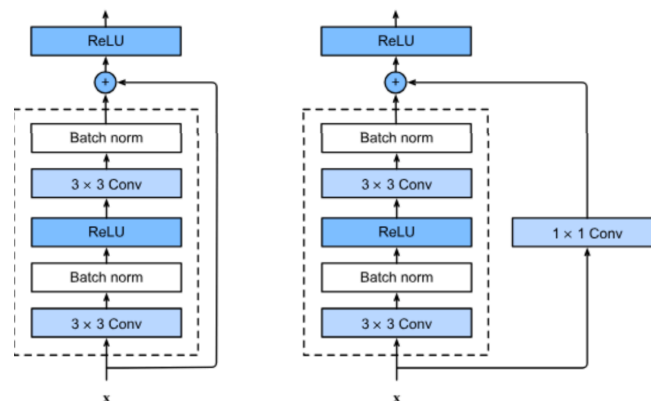
Die Tatsache, dass diese Gradientenoperation mit der Ableitung durchgeführt wird, war das größte Problem bei den vor ResNet verwendeten Modellen. Denn die Ableitungsfunktionen

wurden mit zunehmender Anzahl der Schichten zu kleineren Zahlen. Wenn diese Ableitungen zu einer Zahl im Bereich $[0, 1)$ werden, wird der Gradient fast Null. Da die Gewichte in der ersten Schicht nicht aktualisiert werden konnten, war die Lernleistung in Netzen mit einer hohen Anzahl von Schichten sehr gering. Dieses Problem wird in der Literatur als Gradientenschwund bezeichnet.

Die Struktur, die ResNet Skip Connection nennt, unterbricht die Schrumpfungskette in dieser Ableitungskette. ResNet fügt jedem Block eine Abkürzung hinzu, die als Restverbindung bezeichnet wird. Allerdings lernt das Modell nur die Differenz.

Da der Gradient immer rückwärts durch x fließt, d. h. $d(x)/d(x) = 1$, selbst wenn df/dx sich 0 nähert, ist der Gesamtgradient nicht 0 und es wird immer ein gewisser Gradient übertragen. Dies garantiert das Lernen.

Kurz gesagt, ResNet lernt sowohl genauer als auch schneller, da es sich nur mit dem Lernen neuer Informationen befasst und die richtigen Informationen überspringt und mitnimmt.



Ergebnisse

Vorhersagen:

1. Siamese cat — 97.17%
2. Egyptian cat — 0.69%
3. lynx — 0.15%
4. black-footed ferret — 0.13%
5. tabby — 0.12%

□ Berechnungszeit: 0.43 sekunden

1.b. EfficientNet

Einleitung auf das Modell

EfficientNet-B0 ist ein effizientes Convolutional Neural Network (CNN), das 2019 von Tan und Le bei Google AI entwickelt wurde. Es ist die Grundlage der EfficientNet-Modellfamilie

(B0 bis B7). Das Ziel dieser Familie ist, mit möglichst geringem Rechenaufwand eine hohe Klassifikationsgenauigkeit zu erreichen. Der zentrale Unterschied zu bisherigen Modellen liegt im sogenannten Compound Scaling: Dabei werden die drei Hauptdimensionen eines CNNs – Tiefe (Anzahl der Schichten), Breite (Anzahl der Kanäle) und Auflösung (Größe der Eingabebilder) – nicht unabhängig voneinander, sondern gemeinsam und ausgewogen skaliert. Dieser Ansatz erlaubt es, Modelle proportional zu vergrößern, ohne dass dabei eine der Dimensionen überbetont wird oder die Effizienz leidet.

Architektur und Funktionsweise

Die Architektur von EfficientNet-B0 basiert auf sogenannten MBConv-Blöcken. Dabei handelt es sich um eine strukturell optimierte Variante von Faltungsschichten, die ursprünglich aus MobileNetV2 übernommen wurden. Ein MBConv-Block besteht aus einer initialen Erweiterung der Kanalzahl, einer depthwise-separable Faltung zur Analyse wichtiger Bildstrukturen und einer abschließenden Reduktion auf die ursprüngliche Dimensionalität. Diese Blöcke sind besonders speicher- und rechenzeiteffizient. Das führt zu einer deutlichen Reduktion der Modellkomplexität, ohne die Genauigkeit zu beeinträchtigen. Dies wird durch den Einsatz der Swish-Aktivierungsfunktion anstelle der klassischen ReLU unterstützt. Swish, definiert als $f(x) = x \cdot \text{sigmoid}(x)$, sorgt durch ihren sanfteren Verlauf für stabilere Gradienten während des Trainings. Dies führt vor allem bei tiefen Netzwerken zu einem besseren Lernverhalten.

Theoretischer Hintergrund

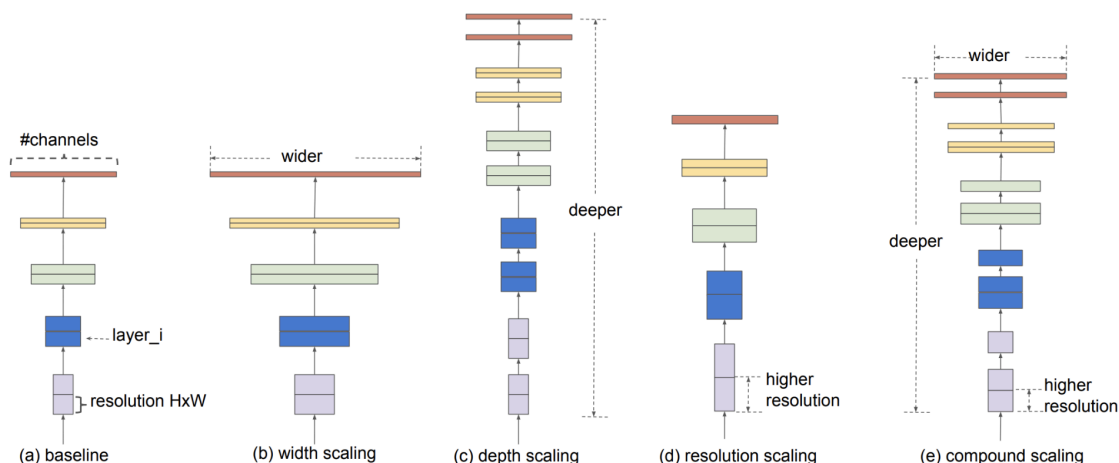


Figure 2. Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

https://production-media.paperswithcode.com/methods/Screen_Shot_2020-06-06_at_10.45.54_PM.png

Abbildung 1: Vergleich von EfficientNets Compound Scaling mit klassischen Skalierungsansätzen.

- (a): Ein grundlegendes CNN-Modell
- (b), (c), (d): Klassische Methoden mit nur erhöhter Breite, Tiefe oder Auflösung.

- (e): Der von EfficientNet vorgeschlagene Compound-Scaling-Ansatz, bei dem alle drei Dimensionen proportional wachsen.

Tiefe = α^ϕ

Breite = β^ϕ

Auflösung = γ^ϕ

ϕ -> Dies ist der Hauptsteuerungsparameter. Er bestimmt, wie groß das Modell skaliert wird. Wenn ϕ größer wird, werden die Anzahl der Schichten, die Anzahl der Kanäle und die Eingabeauflösung des Modells größer.

α -> Dieser Wert legt fest, wie tief das Modell ist. Er bestimmt, wie viele Schichten das Netzwerk hat. Das Modell kann komplexere Muster erkennen, weil es tiefer ist.

β -> Dieser Koeffizient zeigt, wie breit das Modell ist. Er bestimmt die Anzahl der Kanäle (Filter) pro Schicht. Ein größeres Modell kann mehr Merkmale gleichzeitig analysieren.

γ -> Dieser Faktor beeinflusst die Eingabeauflösung. Eine höhere Auflösung bedeutet, dass feinere Details im Bild erfasst werden können. Dafür wird aber auch mehr Rechenaufwand benötigt.

Bildeingabe	Ergebnisse
	<p>Vorhersagen:</p> <ol style="list-style-type: none"> 1. Siamese cat — 89.27% 2. Egyptian cat — 0.84% 3. black-footed ferret — 0.19% 4. polecat — 0.10% 5. indri — 0.09% <p>□ Berechnungszeit: 0.25 sekunden</p>

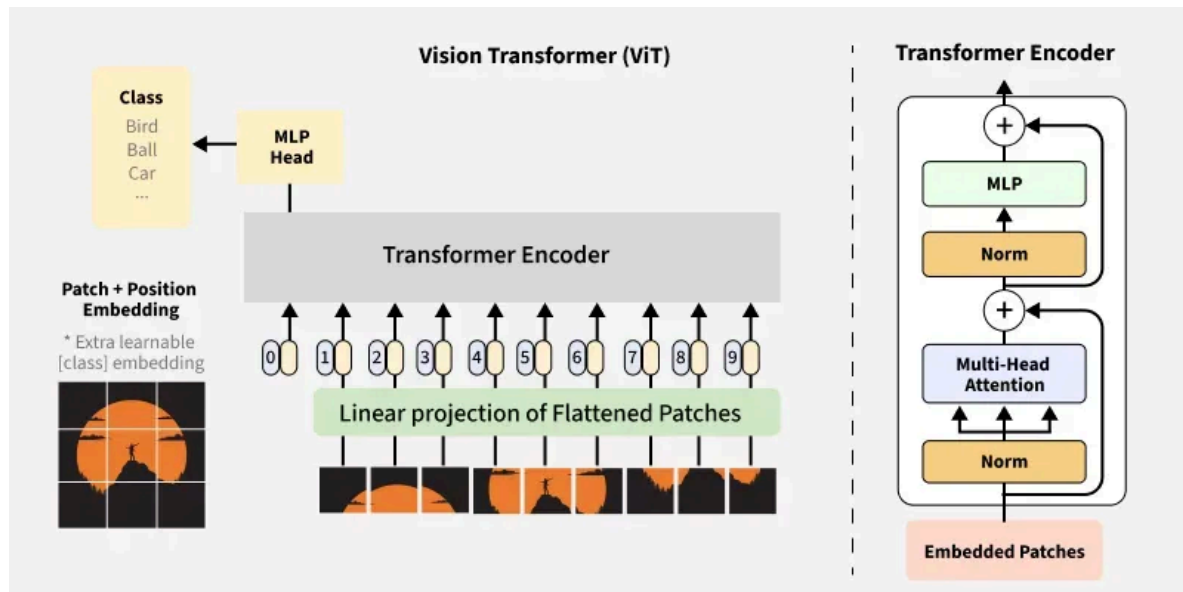
1.c. Virtual Transformer

Einleitung auf das Modell

2020 wurde der Vision Transformer (ViT), ein neuartiges Modell zur Bildklassifikation, von der Google-Brain-Forschungsgruppe präsentiert. Es ist eine entscheidende Weiterentwicklung in der Anwendung von Transformer-Architekturen, die zuvor hauptsächlich im Bereich der natürlichen Sprachverarbeitung (NLP) verwendet wurden. Die in der NLP äußerst erfolgreichen Selbstaufmerksamkeitsmechanismen (Self-Attention)

können auch auf Bilddaten angewendet werden, wie das bahnbrechende Paper „An Image is Worth 16x16 Words“ demonstriert hat.

ViT formt ein Eingabebild in ein Feld fester Patch-Größen (z. B. 16 x 16 Pixel) um, von denen jedes als ein einzelnes Wort behandelt wird. Diese Fleckensymbole werden zusammen mit ihren Standortinformationen in das Transformer-Netzwerk eingefügt, wo ihre globale Beziehung über mehrere Schichten modelliert wird. ViT erlaubt daher eine globale Kontextualisierung in den frühen Phasen des Netzwerks, während traditionelle CNNs normalerweise nur lokale Muster untersuchen.



Theoretischer Hintergrund

Der Vision Transformer basiert auf der klassischen Transformer-Architektur, die eigentlich für Sprachverarbeitungsaufgaben wie die maschinelle Übersetzung entwickelt wurde. Das Herzstück dieses Modells ist die Self-Attention-Methode, mit der Beziehungen zwischen unterschiedlichen Token (in diesem Fall Bildausschnitte) unabhängig von ihrer Position modelliert werden können. Dadurch kann das Modell effizient lernen, welche Teile des Bildes miteinander in Beziehung stehen, was speziell für strukturierte Bilddaten wie Tabellen oder Diagramme wichtig ist.

Aus mathematischer Sicht funktioniert die Selbstaufmerksamkeit, wenn jedem Eingabe-Token drei Vektoren zugeordnet werden: Abfrage (Q), Schlüssel (K) und Wert (V). Die Gewichte der Selbstaufmerksamkeit werden durch das Skalarprodukt zwischen Q und K kalkuliert, durch die Wurzel der Dimension (d_k) normalisiert und dann mit Softmax skaliert:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

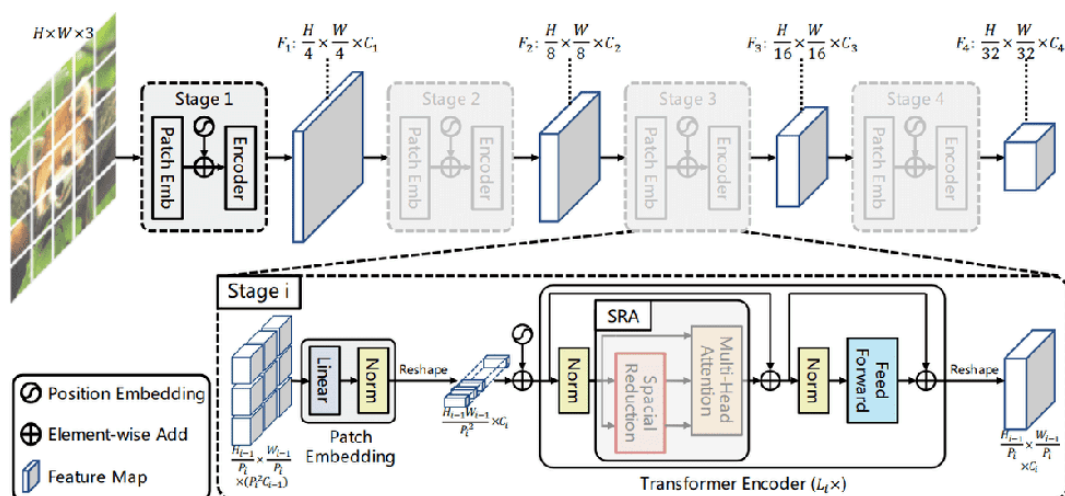
Dieser Vorgang erlaubt es dem Modell, sich dynamisch auf relevante Regionen im Bildkontext zu konzentrieren. Der Transformer setzt sich aus mehreren dieser Schichten

zusammen, kombiniert mit Schichtnormalisierung, Restverbindungen und MLP-Blöcken (Multi-Layer Perceptrons).

Die wichtigsten Modellparameter eines ViT sind:

- Patch-Größe: In der Regel 16x16 oder 32x32 Pixel. Je kleiner, desto feinkörniger die Analyse, aber auch desto größer der Rechenaufwand.
- Dimension der Einbettung: Gibt die Dimension an, in der die Patches eingebettet sind (z. B. 768 für ViT-Base).
- Tiefe: Anzahl der zusammenhängenden Transformatorenblöcke (z. B. 12 für ViT-Base).
- Anzahl der Aufmerksamkeitsköpfe: Gibt die Parallelisierung in der Selbstaufmerksamkeit an (z. B. 12 für ViT-Base).
- MLP-Dimension: Größe der versteckten Schicht im Feed-Forward-Netz innerhalb jedes Blocks.
- Positionskodierung: Erforderlich, da der Transformator selbst keine inhärente Reihenfolge kennt.

Eine Herausforderung bei der Anwendung von ViT ist der Bedarf an großen Datenmengen für ein erfolgreiches Training. Daher werden häufig vortrainierte Modelle benutzt (z. B. ImageNet-21k oder JFT-300M). Für unser Projekt bedeutet dies, dass wir ein fein eingestelltes ViT-Modell verwenden, das bereits grundlegende visuelle Konzepte gelernt hat und nur noch für unsere spezifischen Klassen angepasst werden muss.



Bildeingabe



Ergebnisse

Vorhersagen:

1. Siamese cat — 86.07%
2. Egyptian cat — 0.94%
3. window screen — 0.05%
4. lynx — 0.05%
5. Australian terrier — 0.05%

□ Berechnungszeit: 0.97 sekunden

1. Vergleich der Modelle

2.a. Prüfung und Vergleich von Modellen

ResNet, EfficientNet und ViT. Da alle drei Modelle auf dem ImageNet-Datensatz trainiert wurden, beschlossen wir, diese drei Modelle mit ihren vortrainierten Versionen zu vergleichen. Zu diesem Zweck haben wir 5 kürzlich aus dem Internet hochgeladene Bilder ausgewählt, bei denen es sehr unwahrscheinlich ist, dass sie für das Training des Modells verwendet werden. Diese Bilder sind Golden Retriever, Zebra, Pizza, Laptop und Banane.



Wir haben die Modelle anhand der folgenden Kriterien verglichen:

- **Ihre Vorhersage**
- **Genauigkeit**
- **die Zeit, die sie für den Prozess aufwenden**

	Bild	ResNet50 Vorhersage	ResNet50 Genauigkeit	ResNet50 Zeit	EfficientNet- B0 Vorhersage	EfficientNet- B0 Genauigkeit	EfficientNet- B0 Zeit	Vision Transformer Vorhersage	Vision Transformer Genauigkeit	Vision Transformer Zeit
0	banana.jpeg	banana	60.42%	0.296 Sekunden	banana	18.20%	0.178 Sekunden	banana	41.62%	0.544 Sekunden
1	goldenretriever.jpg	golden retriever	91.63%	0.253 Sekunden	golden retriever	87.34%	0.146 Sekunden	golden retriever	47.49%	0.526 Sekunden
2	laptop.jpg	notebook	53.82%	0.245 Sekunden	desktop computer	20.01%	0.138 Sekunden	desktop computer	24.03%	0.534 Sekunden
3	pizza-tonno.jpg	pizza	99.68%	0.250 Sekunden	pizza	89.49%	0.147 Sekunden	pizza	90.29%	0.540 Sekunden
4	zebra.jpg	zebra	100.00%	0.250 Sekunden	zebra	92.05%	0.139 Sekunden	zebra	94.27%	0.510 Sekunden

(Der Quellcode ist in den Dateien zu finden)

Alle Vorhersagen von ResNet50 sind recht vernünftig und sehr genau. Anhand dieses kleinen Tests lässt sich sagen, dass ResNet50 ein stabileres Modell ist. EfficientNet-B0 machte eine überraschend niedrige Vorhersage von 18,2 % für das Bananenbild. Trotz des geringen Vertrauens in einigen Klassen kann man sagen, dass das Modell genaue Vorhersagen macht. ViT hingegen macht zwar ebenso genaue Vorhersagen wie seine Konkurrenten, aber sein niedriger Konfidenzwert ist auffällig. Betrachtet man außerdem die Vorhersagezeiten aller Modelle, so wird deutlich, dass ViT im Vergleich zu seinen Konkurrenten recht langsam ist. EfficientNet-B0 ist seinen Konkurrenten in Bezug auf die Vorhersagezeit voraus.

2.b. Bewertung und Entscheidung für ein Modell

Bei der Auswahl des Modells müssen die Anforderungen des AI-Document-Checker-Projekts berücksichtigt werden. Der im Projekt bereits vorhandene Datensatz enthält eine relativ geringe Menge an gelabelten Daten. Hier kommt EfficientNet ins Spiel. Da EfficientNet im Gegensatz zur traditionellen Methode ein "compound scaling" bietet, vergrößert es die Dimensionen Tiefe-Breite-Auflösung proportionaler und ausgewogener.

Darüber hinaus lieferte EfficientNet(-B0) mit 5 Millionen Parametern ähnliche oder sogar bessere Ergebnisse in ImageNet als das ResNet50-Modell mit der fünffachen Anzahl von Parametern. Mit anderen Worten: Es ist möglich, mit weniger Berechnungen, weniger Arbeitsspeicher und weniger Zeit bessere Ergebnisse zu erzielen.

Zusätzlich bieten die MBConv-Blöcke in der EfficientNet-Struktur und die verwendete Swish-Aktivierung schnellere Schichten und eine bessere Optimierung. Diese optimierte Architektur ermöglicht den Einsatz von EfficientNet auf mobilen/Edge-Geräten.

Da die Ergebnisse unseres Projekts in Prozessen wie der **automatischen Klassifizierung von Bildern, der Generierung von JSON-Ausgaben, der Bewertung und dem Vergleich** verwendet werden, muss das trainierte Modell **kontinuierlich** genutzt werden. Dies macht **eine schnelle Verarbeitungszeit** noch wichtiger. Mit diesen und den vorangegangenen Erklärungen kamen wir zu dem Schluss, dass das EfficientNet-B0-Modell die richtige Wahl für das Lernen des Datensatzes in unserem Projekt wäre.

2. Training des ausgewählten Modells

2.a Erkennen und Organisieren des Datensatzes

Neben dem Erkennen der Anforderungen des von uns verwendeten Modells ist es von entscheidender Bedeutung, den Datensatz zu erkennen, um den vorhandenen Datensatz beim Training des Modells nach Möglichkeit besser zu nutzen und zukünftige Vorhersagen von besserer Qualität zu treffen. Da es sich bei diesem Prozess um eine Aufgabe handelt,

die zu Beginn unseres Arbeitsablaufs abgeschlossen werden muss, ist es außerdem wichtig, darauf zu achten, dass wir in den folgenden Schritten nicht zurückgehen müssen.

Einer der wichtigsten Punkte beim Training eines Modells ist die Qualität des verwendeten Datensatzes. Bei der Untersuchung der uns zur Verfügung stehenden Daten zeigt sich, dass der Datensatz aus zwei getrennten Dateien besteht, nämlich aus gelabelten und nicht gelabelten. Da es sich bei EfficientNet-B0 um ein CNN handelt, wird für die Klassifizierung die Methode des überwachten Lernens verwendet. Aus diesem Grund ist es klar, dass wir für das Training gelabelte Daten verwenden werden. Wenn wir uns die uns zur Verfügung gestellten gelabelten Daten ansehen, sehen wir, dass sie aus 13 verschiedenen Dateien, d.h. Labels/Klassen, bestehen. Jede Klasse hat eine unterschiedliche Anzahl von Daten. An dieser Stelle sollte man bedenken, dass die Menge der Daten zwar wichtig für das Training ist, es aber wichtiger ist, dass die Daten eine hohe Qualität haben, was die Qualität betrifft. Mit anderen Worten, es ist wichtiger, dass das Bild, das zu dieser Klasse gehört, mehr wünschenswerte Merkmale usw. enthält, damit das Modell lernen kann. Zunächst haben wir unsere Daten von 13 Klassen auf 6 Klassen reduziert, aber für einige Klassen wurde eine Generalisierung vorgenommen und wir haben bestimmt, welche Klassen zu den neuen 6 Klassen gehören. Diese Zuordnungen sind in *label_mapping.py* zu sehen.

Unter Verwendung dieser Zuordnungsstruktur wird die Struktur in *mapper.ipynb* verwendet, um alle Daten gemäß der neuen Struktur zu klassifizieren. So werden die Daten mit Tags nach den folgenden Klassen umverteilt: Excel-Tabelle, Datenfluss, Info-Objekt, Transformation, Datentransferprozess, Datenquelle.

Unser Datensatz besteht aus Excell und Screenshots aus dem SAP-Programm.

2.b Vorbereitung des Modells für das Training

Wie bereits erwähnt, enthalten die Klassen des Datensatzes nicht die gleiche Anzahl von Daten. Während die Klasse "Excel-Tabelle" beispielsweise 132 visuelle Daten enthält, enthält unsere Klasse "Datenfluss" nur 44 visuelle Daten. Da dies den Erfolg des Modells beim Training und die Vorhersagen in den nächsten Schritten unseres Projekts beeinflussen kann, haben wir eine Datenaugmentation vorgenommen.

Der Grund für die Datenanpassung:

1. Unser Datensatz ist klein

Unser Datensatz besteht also nicht aus Millionen von Bildern. Weniger Daten können dazu führen, dass sich das Modell einprägt, d. h. dass es sich zu stark anpasst. Durch die Erweiterung kann das Modell mehr Bilder sehen.

2. Das Modell haltbarer machen

In realen Beispielen werden die Daten nicht immer ideal dargestellt. Mit anderen Worten, wenn wir Beispiele aus unserem Projekt nehmen, widerspricht es dem üblichen Lebensfluss, von den Studierenden zu erwarten, dass sie das Bild im gleichen Winkel und in der gleichen Größe wie in der Musterlösung senden. Aus diesem Grund bereiten wir das Modell auf die reale Welt vor, indem wir leicht verzerrte Bilder zeigen.

Ein Blick auf die Datei *model_training.ipynb* zeigt, wie die Augmentation in der **Transforms-Phase** des Codes durchgeführt wird. Demnach wird das Bild, nachdem die von

EfficientNet-B0 erwarteten Bildabmessungen eingestellt wurden, **horizontal gespiegelt, zufällig um 10 Grad gedreht und mit seiner Helligkeit und seinem Kontrast gespielt**. Dies bedeutet, dass das Modell in jeder Epoche unterschiedliche Variationen desselben Bildes sieht. Wenn zum Beispiel x Epochen stattgefunden haben, hat das Modell x Variationen desselben Bildes gesehen und gelernt. Kurz gesagt, die Augmentation wird bei jedem Trainingsschritt (Epoche) nach dem Zufallsprinzip angewandt und die Leistung des Modells im realen Leben wird erhöht.

2.c Modelltraining und Ergebnisse

Für das Training des Modells haben wir das vortrainierte Modell von EfficientNet-B0 mit Gewichten verwendet, die auf ImageNet trainiert wurden, um unseren eigenen Datensatz zu trainieren, d.h. Transfer-Lernen, so dass das Modell bereits die grundlegenden Bildmerkmale kennt und wir es leicht an unseren eigenen anpassen können. Einzelheiten sind in der Trainingsdatei des Modells zu finden (*model_training.ipynb*).

Wir haben auch versucht, den Trainingserfolg mit der von uns verwendeten Trainingsmethode zu maximieren, d.h. während des Trainings des Modells werden am Ende jeder Epoche sowohl train loss, validation loss und validation accuracy berechnet. Wenn sich die **Validierungsgenauigkeit über mehrere Epochen hinweg nicht signifikant verbessert**, wird das Training beendet. Mit anderen Worten, wenn es für 3 Epochen keine Verbesserung gibt, wird das Training des Modells gestoppt, dies wird als 'early stopping' bezeichnet. Es wird erwartet, dass ein minimaler Anstieg von 0,1 % als Verbesserung gewertet wird. Mit dieser Logik wollten wir Trainingszeit sparen, so dass die Anzahl der Epochen nicht im Voraus eingegeben werden muss. Die Überanpassung nimmt ab, denn in dem Szenario, in dem die Genauigkeit der Modellvalidierung nicht zunimmt, das Training aber fortgesetzt wird, wird das Training vergeblich fortgesetzt und das Modell beginnt, sich zu merken und die Überanpassung nimmt zu. Mit dieser Logik können wir das beste Modell erreichen.

Die Entwicklung des Modells in der Trainingsphase ist unten zu sehen.

```
Epoch 1: Train Loss = 1.6116, Val Loss = 1.4781, Val Accuracy = 58.25%
Epoch 2: Train Loss = 1.1505, Val Loss = 0.9655, Val Accuracy = 72.82%
Epoch 3: Train Loss = 0.8166, Val Loss = 0.6920, Val Accuracy = 77.67%
Epoch 4: Train Loss = 0.5842, Val Loss = 0.5270, Val Accuracy = 80.58%
Epoch 5: Train Loss = 0.4021, Val Loss = 0.4086, Val Accuracy = 87.38%
Epoch 6: Train Loss = 0.2558, Val Loss = 0.2998, Val Accuracy = 91.26%
Epoch 7: Train Loss = 0.1764, Val Loss = 0.2188, Val Accuracy = 93.20%
Epoch 8: Train Loss = 0.1101, Val Loss = 0.1743, Val Accuracy = 93.20%
⚠ No significant improvement for 1 epoch(s).
Epoch 9: Train Loss = 0.0832, Val Loss = 0.1569, Val Accuracy = 93.20%
⚠ No significant improvement for 2 epoch(s).
Epoch 10: Train Loss = 0.0560, Val Loss = 0.1514, Val Accuracy = 94.17%
Epoch 11: Train Loss = 0.0439, Val Loss = 0.1455, Val Accuracy = 94.17%
⚠ No significant improvement for 1 epoch(s).
Epoch 12: Train Loss = 0.0506, Val Loss = 0.1475, Val Accuracy = 93.20%
⚠ No significant improvement for 2 epoch(s).
Epoch 13: Train Loss = 0.0385, Val Loss = 0.1479, Val Accuracy = 94.17%
⚠ No significant improvement for 3 epoch(s).
!!!Early stopping triggered.
```

in model_training.ipynb zu finden

Eine ImageClassifier-Klasse (image_classifier.py) wurde geschrieben, um sie in den späteren Phasen des Projekts dynamischer nutzen zu können.

Ein kleines Beispiel:

Für das Beispiel verwendetes Bild:

The screenshot shows the SAP Analytics Cloud interface. The main table displays product categories and their associated financial data. The table has columns for Product Category, Product, Revenue, Discount, and Percentualer Anteil. The data is filtered for the year 2014 and the country Germany. The table shows a total revenue of 16,086,764.60 and a total discount of 511,666.19.

Product Category	Product	Revenue	Discount	Percentualer Anteil
Accessories	Air Pump	120,264.04	3,940.11	3.28
	Elbow Pads	28,905.41	841.60	2.91
	First Aid Kit	70,260.04	2,202.70	3.14
	Knee Pads	33,083.46	1,060.72	3.21
	Off Road Helmet	46,100.87	1,443.41	3.13
	Repair Kit	27,030.34	862.23	3.19
	Road Helmet	43,258.03	1,412.72	3.27
	T-shirt	25,545.90	847.65	3.32
	Water Bottle	17,371.65	551.79	3.18
	Water Bottle Cage	46,146.29	1,472.16	3.19
Result		457,967.23	14,635.08	3.20
E Bikes	E-Bike Tailwind	3,506,168.76	116,008.21	3.31
Result		3,506,168.76	116,008.21	3.31
Offroad Bikes	Men's Off Road Bike Fully	5,770,853.13	183,930.43	3.19
	Men's Off Road Bike Hard Tail (Shimano)	2,793,058.00	91,114.90	3.26
	Men's Off Road Bike Hard Tail (SRAM)	4,667,501.60	148,142.40	3.17
	Women's Off Road Bike Fully	2,850,751.27	88,478.46	3.10
Result		16,086,764.60	511,666.19	3.18

Output:

Prediction: Excel-Tabelle
Confidence: 0.997

3: Multimodales Large Language Model

3.a Definition

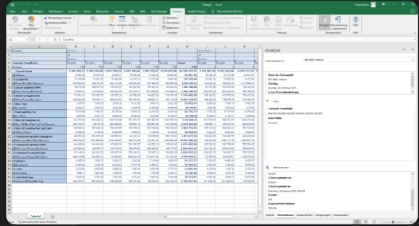
Ein multimodales Large Language Model (MLLM) ist ein KI-System, das neben Text auch andere Datenformen, insbesondere Bilder, verarbeiten kann. Im Gegensatz zu klassischen Large Language Models, die ausschließlich mit sequenziellen Texteingaben arbeiten, erweitert ein MLLM seine Fähigkeiten um einen visuellen Encoder (z. B. einen Vision Transformer oder ein CNN). Dadurch ist es in der Lage, Bildinhalte gemeinsam mit Textanfragen zu verstehen und zu beantworten.

3.b Prompting mit multimodalen Modellen

Ein wichtiger Vorteil multimodaler Sprachmodelle liegt im sogenannten Prompting: Durch gezielte Anfragen in natürlicher Sprache kann man das Verhalten des Modells steuern, ohne es extra trainieren zu müssen. Das Modell interpretiert dabei nicht nur die sprachliche Eingabe, sondern berücksichtigt auch die visuelle Information im Prompt-Kontext. Beispiele für die Nutzung sind:

- **Bildbeschreibung:** „Beschreibe in einem Satz, was dieses Diagramm zeigt.“
- **Objekterkennung:** „Zähle alle erkennbaren Tabellen auf diesem Screenshot.“
- **Diagramm-Analyse:** „Vergleiche dieses Data-Flow-Diagramm mit der Referenz und nenne Abweichungen.“
- **Multimodale Q&A:** Bild + Frage „Welcher Prozessschritt folgt nach dem Filter?“

ChatGPT 4o ↑ Gemeinsam nutzen



Analysiere diesen Screenshot und erkläre die Inhalte

Dieser Screenshot zeigt eine geöffnete Excel-Datei mit einer Pivot-ähnlichen Analyseansicht in Microsoft Excel, vermutlich aus einem **SAP-Umfeld oder BI-Tool**, da oben die Registerkarten „EPM“, „Data Manager“, „Analysis Design“ usw. sichtbar sind. Es handelt sich um eine Datenanalyse im Rahmen eines **Vertriebs- oder Umsatzberichts**.

↓

4. Bewertungssystem für visuelle Zustellungen mit einem multimodalen LLM

Im Rahmen dieser Arbeit wurde ein vollständiger und skalierbarer Bewertungsansatz entwickelt, um visuelle Zustellungen automatisch zu analysieren und - sofern vorhanden - mit einer Musterlösung zu vergleichen. Dieser Abschnitt beschreibt den technischen Aufbau, die Auswertungslogik und die Integration in die Gesamtarchitektur.

4.a Technischer Aufbau des Bewertungssystems

Für die Bewertung wurde das multimodale LLM Qwen-VL-Chat verwendet. Dieses Modell bietet eine optimale Kombination aus Geschwindigkeit, Ressourcenverbrauch und Modellgröße. Im Vergleich zu größeren Varianten (wie z.B. Qwen-VL-Max) war es deutlich schneller, brauchte weniger Speicherplatz und konnte auf der gegebenen Serverinfrastruktur problemlos eingesetzt werden. Gleichzeitig erwies sich die Qualität der Ausgabe für den vorgesehenen Zweck als völlig ausreichend.

Ein besonderes Problem entstand beim Hochladen von zwei getrennten Bildern (Vorlage + Musterlösung), da einige Modelle diese nicht zuverlässig zusammen verarbeiten konnten. Um dieses Problem zu lösen, wurden beide Bilder lokal zusammengeführt und als ein einziges zusammengesetztes Bild im Base64-Format an das Modell übertragen.

Darüber hinaus wurde ein Bewertungsprozess entwickelt, der zwischen zwei Anwendungsfällen differenziert:

1. Mit Musterlösung: Wenn der Benutzer eine Musterlösung einreicht, wird diese verwendet, um die Abgabe visuell und strukturell direkt zu vergleichen.
2. Ohne Musterlösung: Liegt keine Referenz vor, wird die Einsendung automatisch klassifiziert (z.B. „Datenfluss“) und mit bestehenden Beispielen in der Datenbank verglichen. In diesem Fall generiert das Modell eine Bewertung auf der Grundlage der vorliegend ähnlichsten Lösung dieser Klasse.

4.b Strukturiertes Scoring mit JSON-Ausgabe

Das Modell wurde angewiesen, seine Scores in einer strukturierten JSON-Ausgabe zu liefern. Für jede Klasse (z.B. Excel-Tabelle, Datenflussdiagramm, DTP-Schema, Transformation) wurden individuelle, präzise Prompts entwickelt, die eine standardisierte Bewertung basierend auf mehreren Dimensionen ermöglichen. Jedes dieser Prompts führte zu einer Bewertung entlang mehrerer Kriterien, darunter Strukturqualität, visuelle Gestaltung, technische Korrektheit, SAP-Kontext und Verständlichkeit. Jede Dimension wurde auf einer Skala von 0 bis 10 bewertet. Zusätzlich wurde ein wertvolles Feedback in Form von Feedback-Vorschlägen generiert. Diese JSON-Ausgaben bilden die Grundlage für objektive und vergleichbare Auswertungen.

4.c Vergleich mit der Musterlösung

Wenn der Benutzer eine Musterlösung bereitstellt, wird auch eine Vergleichsabfrage verwendet. Diese wurde speziell entwickelt, um die visuelle und strukturelle Übereinstimmung zwischen der bereitgestellten Lösung und der Referenzlösung zu bewerten. Das Modell beachtet die folgenden Kriterien:

- Strukturelle Ähnlichkeit
- Inhaltliche Korrektheit
- Technische Umsetzung
- Vollständigkeit

- Visuelle Klarheit

Beide Bilder werden von dem Modell analysiert und gemäß diesen Merkmalen verglichen. Das Ergebnis ist eine Gesamtnote zwischen 0 und 100 sowie eine Note (1,0-5,0). Wenn dieser Modus aktiviert ist, geht die vergleichende Bewertung zu 50% in die Gesamtbewertung ein, während die restlichen 50% durch allgemeine Bewertungskriterien bestimmt werden.

4.d Automatisierte Erzeugung von Metadaten

Die vom Modell produzierten Bewertungsergebnisse wurden automatisch in strukturierte Metadaten-Dateien umgewandelt. Für jedes analysierte Bild wurde eine JSON-Datei erstellt, die neben der Gesamtbewertung auch die Teilbewertungen für die einzelnen Kriterien und erklärende Rückmeldungen enthält.

Diese Dateien enthalten Felder wie:

- Bildname
- Klasse (z. B. „Excel“, „DTP“)
- Gesamtpunktzahl
- Kritiker_Bewertungen
- Rückmeldung

Diese Dateien wurden systematisch dem jeweiligen Bild zugeordnet und gespeichert, so dass eine maschinenlesbare Grundlage für individuelle Auswertungen und weitere Analysen erstellt wurde.

Wenn zu einer Einsendung auch eine Musterlösung vorlag, wurde auch ein direkter Vergleich zwischen den beiden Bildern durchgeführt. Die Ergebnisse dieses Vergleichs - wie Unterschiede in Struktur, Inhalt oder Darstellung - wurden ebenfalls im JSON-Format dokumentiert und zusammen mit dem jeweiligen Bild gespeichert.

Auf diese Weise wurde der Bewertungsprozess vollständig digitalisiert und ermöglicht eine homogene, transparente und skalierbare Bewertung aller Einreichungen.

4.e Fazit zur Aufgabenstellung

Der im Rahmen dieser Arbeit entwickelte Evaluierungsansatz erfüllt die Anforderungen der Aufgabe 4 vollständig. Es wurde ein multimodales großes Sprachmodell verwendet, das in der Lage ist, visuelle Inhalte strukturiert zu analysieren und zu bewerten. Für jede Bildklasse wurden konkrete und präzise Prompts formuliert, die eine differenzierte Bewertung der Einsendungen ermöglichen. Die Ergebnisse dieser Bewertungen werden in einer

standardisierten JSON-Struktur ausgegeben, die sowohl numerische Teilbewertungen als auch qualitatives Feedback enthält.

Darüber hinaus wurde eine spezielle Abfrage entwickelt, um die übermittelten Bilder mit einer Musterlösung zu vergleichen. Dieser Vergleich berücksichtigt sowohl strukturelle als auch inhaltliche Aspekte und fließt - sofern vorhanden - in die Gesamtbewertung ein. Die generierten Bewertungen werden systematisch als Metadaten-Dateien gespeichert, so dass sie den jeweiligen Einreichungen eindeutig zugeordnet und später weiterverarbeitet bzw. analysiert werden können.

Insgesamt handelt es sich um ein konsistentes, objektives und einleuchtendes Bewertungssystem, das sowohl für ein individuelles Feedback als auch für eine übergreifende Qualitätssicherung und statistische Auswertung geeignet ist.

5. Bewertung der Lösung

Der von uns entwickelte Ansatz beinhaltet detaillierte Eingabeaufforderungen. Unser Ziel ist es jedoch, die Grenzen des von uns verwendeten MLLM-Modells so weit wie möglich zu bestimmen und im Detail zu verarbeiten, dass es bei seinen Vergleichen nur die von uns angegebenen Merkmale berücksichtigt und in welchem Format es die Ausgabe machen soll usw. Doch selbst wenn wir versuchen, detailliert zu arbeiten und Grenzen zu ziehen, liefert das Modell nicht immer genaue Ergebnisse. So wurde z. B. beobachtet, dass das Modell selbst dann, wenn es nachdrücklich aufgefordert wurde, die 50%ige Punktzahl durch einen Eins-zu-Eins-Vergleich zu ermitteln, diese Anforderung nicht vollständig erfüllte. Vor allem bei Musterlösungen und Abgaben kann es sein, dass das Modell je nach Komplexität des Bildes, genauer gesagt, je nach Fülle des Bildinhalts, die Erwartungen bei der Durchführung von Vergleichen nicht erfüllt. Zum Beispiel kann das Modell impulsiver sein, wenn die Antworten in der abgabets nicht der Erwartung der musterlösung entsprechen. Da es beim Vergleich von strukturierten Daten wie z.B. Excell-Tabelle, Data-Flow einfacher ist, dem Modell innerhalb der Abfrage Grenzen zu ziehen, kann man sagen, dass das Modell beim Vergleich solcher visuellen Klassen erfolgreicher ist. Bei der Bewertung dieser und anderer Klassen außerhalb dieser Struktur wird es schwierig, Grenzen zu ziehen, und daher wird es schwieriger, einen allgemeinen Rahmen zu schaffen. Obwohl das Bewertungssystem unseres Ansatzes zufriedenstellend und logisch ist, ist die Tatsache, dass das Modell manchmal unterschiedliche Bewertungen für dieselben Eingaben vergibt, ein Fragezeichen. Dies ist wiederum auf die Schwierigkeit zurückzuführen, unsere Kategorien in der

Eingabeaufforderung in einen allgemeinen Rahmen einzupassen. Mit anderen Worten, das Modell kommt innerhalb der Grenzen, die ihm bei der Bewertung gesetzt wurden, oft zu unterschiedlichen Schlussfolgerungen. Zum Beispiel gibt das Modell bei einem Vergleich der Anzahl der Zeilen für die gleichen Beispiele unterschiedliche Werte an, die nahe beieinander liegen, wie z. B. 3-4, wenn eine Punktzahl zwischen 0-5 Punkten eingegeben wird. Es ist also unbestreitbar, dass das Modell seine eigene Meinung hat. Bei einer allgemeinen Bewertung funktioniert unser Ansatz jedoch erfolgreich, und es ist nachvollziehbar, auf der Grundlage welcher Kriterien das Modell seine Bewertungen abgibt.

6. Feedback-Generierung mittels Sprachmodell

6.a Einführung

In Aufgabe 6 erweitern wir den bestehenden Evaluierungsworkflow um eine inhaltliche Rückmeldung zu jedem Abgabebild. Neben der visuellen Gegenüberstellung wird für alle relevanten Abgabe-Bilder eine strukturierte JSON-Metabeschreibung erstellt. Anschließend werden die Bilder mit der jeweiligen Referenzlösung verglichen. Das genaue Punkteergebnis wird direkt nach dem Upload angezeigt. Die Gesamtbewertung berücksichtigt nur gültige Paare, während nicht referenzierte Bilder ignoriert werden.

6.b Konzept

6.b.1 Bildextraktion und Vorverarbeitung

Zunächst liest `PDFImageExtractor.extract_images_from_pdf` (Datei `pdf_processor.py`) alle Screenshots aus der Abgabe-PDF aus und speichert sie temporär. Anschließend klassifiziert `ImageClassifier.predict_from_base64` (Datei `image_classifier.py`) jedes Bild in eine der sechs SAP-Kategorien und liefert dazu einen Confidence-Wert. Bilder mit einem Confidence unter 60 % werden aus dem weiteren Prozess ausgeschlossen. Sobald die Kategorie feststeht, prüft `QwenClient.check_image_evaluability` (Datei `qwen_client.py`), ob das Bild frei von störenden UI-Elementen, Fehlermeldungen oder Unschärfen ist. Nur Bilder, die alle drei Filter erfolgreich passieren, werden inhaltlich analysiert.

6.b.2 Erzeugung strukturierter JSON-Metadaten

Für jedes validierte Bild löst die EvaluationEngine einen Aufruf von QwenClient.extract_metadata (ebenfalls in qwen_client.py) aus. Dabei verwendet der Client die kategoriespezifischen Prompt-Vorlagen aus der Datei metadata_templates.py, um Merkmale wie Knotenzahl, Spaltenanzahl, Vorhandensein von Formeln oder SAP-Elemente zu extrahieren. Das Resultat ist stets ein streng formatiertes JSON-Objekt – zum Beispiel enthält ein Data-Flow-JSON Felder wie struktur.knoten_anzahl, technik.pfeile_klar und sap_elemente.transformation. Die detaillierten Prompts sind in evaluation_templates.py zu sehen. Dieses Format stellt sicher, dass alle relevanten Eigenschaften eindeutig abbildbar sind.

6.b.3 Referenz-Metadatenabgleich

Die zuvor mit MetadataGenerator.generate_full_database (Datei metadata_generator.py) erzeugten JSON-Metadaten der Musterlösungen werden in der Engine geladen (Methode _load_metadata_database in evaluation_engine.py). Mittels der Funktion EvaluationEngine._find_top_reference_matches vergleicht die Engine die JSON-Struktur des Abgabe-Bildes mit allen Referenz-JSONs derselben Kategorie. Hierbei wird für jedes Schlüssel-Wert-Paar ein einfacher Übereinstimmungswert berechnet, und das Referenzdokument mit dem höchsten Score wird ausgewählt. Wenn ein Referenzdatensatz nicht ausreichend mit dem Bild übereinstimmt, überspringt die Engine das Bild vollständig. Hierzu wird der unten genannte Ausdruck in den Prompt eingegeben. Also wird das Bild nicht in die Punkteberechnung einbezogen.

WICHTIG: Prüfe zuerst die Bildqualität! Falls das Bild zu schlecht ist (z.B. Rechtsklick-Menü sichtbar, völlig unscharf, falscher Inhalt, nur Fehlermeldungen), setze "skip_evaluation": true.

6.b.4 Detaillierte Bewertung und Feedback

Für jedes erfolgreich gematchte Bild-Referenz-Paar kombiniert QwenClient.detailed_evaluation (ebenfalls in qwen_client.py) diese beiden Bilder zu einer Side-by-Side-Ansicht und sendet sie zusammen mit einem kategoriespezifischen Bewertungs-Prompt Qwen-API-Server. Als Antwort liefert das Modell ein umfassendes Feedback-JSON (vorformuliert in evaluation_templates.py bzw. custom_evaluation_templates.py), das für definierte Subkriterien Punktwerte zwischen 0 und 25 enthält, eine Gesamtpunktzahl (0–100) ausgibt und zusätzlich textuelle Hinweise zu Stärken und Verbesserungspotenzial bereitstellt. Durch diese inhaltliche Tiefe wird ein hochdetailliertes, lernförderndes Feedback ermöglicht.

6.b.5 Berechnung des Gesamt-Scores und Anzeige

Nach Abschluss aller Detailbewertungen sammelt die `EvaluationEngine` in `evaluate_pdf_submission` alle in den Feedback-JSONs unter `gesamt_bewertung.erreichte_punkte` abgelegten Einzelwertungen und berechnet den arithmetischen Mittelwert. Ein Schwellenwert von 70 Punkten entscheidet über Pass/Fail. Abschließend formatiert `EvaluationEngine.get_evaluation_summary` das Ergebnis in eine lesbare Übersicht. Diese enthält die Anzahl und die Bezeichnungen der bewerteten Bilder, ihre Einzelpunktzahlen sowie den Gesamt-Score mit einer „Bestanden“-Kennzeichnung.

7. Frontend für automatisierte Bildbewertung

7.a Einleitung

In Aufgabe 7 erstellen wir eine Web-Benutzeroberfläche, die es erlaubt, eine ZIP-Datei mit SAP-Bilder (bzw. Screenshots) hochzuladen und automatisch bewerten zu lassen. Das Frontend verbindet den Nutzer mit dem Python-Backend (Flask-Server). Die Bedienung soll einfach sein, es sollen klare Rückmeldungen erfolgen und die Ergebnisse sollen übersichtlich dargestellt werden.

7.b Hauptfunktionen

- **Dateiupload**
Nutzer können entweder eine ZIP-Datei (mit einer PDF oder mehreren Bildern) hochladen oder per klassischer Dateiauswahl einzelne Dateien auswählen.
- **Moduswahl**
 - Über eine Checkbox schaltet man zwischen:
 - Standard-Modus (Nutzung der zentralen Referenzdatenbank)
 - Custom-Modus (eigene Referenz-ZIP/PDF/Bilder)
- **Validierung**
Eingaben werden clientseitig geprüft:
 - Zulässige Dateitypen (.zip, .pdf, .jpg, .png)
 - Maximale Dateigröße (10 MB)
 - Maximal fünf einzelne Bilder pro Custom-Referenz

- **Bewertungsanforderung**

Nach erfolgreicher Validierung und Klick auf „Bewertung starten“ sendet das Frontend per `fetch('/api/evaluate')` ein `FormData`-Objekt an den Flask-Endpoint.

- **Lade- und Statusanzeige**

Während der Server arbeitet, zeigt der Evaluate-Button ein Spinner-Icon und „Bewertung läuft...“ an. Bei Fehlern erscheinen Alert-Meldungen.

7.b.1 Referenz-Musterlösung vs. eigene Uploads

Im Standard-Modus ist bereits eine komplette Referenzdatenbank vorhanden, die beim Start von `EvaluationEngine()` in `evaluation_engine.py` aus `metadata_database.json` geladen wird. Jede Abgabe wird dann automatisch gegen diese vordefinierten Musterlösungen ausgewertet.

Im Custom-Modus lädt der Student eigene Referenzdateien hoch (ZIP, PDF oder einzelne Bilder). Die Dateien werden von der Flask-Route `/api/evaluate` mit der Hilfsfunktion `process_custom_references()` (in `app.py`) verarbeitet.

1. ZIP-Archive entpackt oder PDFs in einzelne Screenshots wandelt (`PDFImageExtractor`),
2. jedes Bild per `EfficientNet` klassifiziert (`ImageClassifier.predict_from_base64`),
3. über `QwenClient.extract_metadata()` strukturierte JSON-Metadaten zu jeder Referenz erzeugt (nach Vorgaben aus `metadata_templates.py`).

Dann wird in `process_custom_references()` eine temporäres `metadata_db` erstellt. Diese ersetzt die Original-Datenbank in `EvaluationEngine.metadata_db`, solange die Bewertung läuft. Die Methode `evaluate_pdf_submission(..., custom_mode_only=True)` (in `evaluation_engine.py`) sorgt dafür, dass nur Kategorien ausgewertet werden, für die auch eine eigene Referenz-Beschreibung existiert. Nach Abschluss wird die Original-Referenzdatenbank automatisch wiederhergestellt. So kann der gleiche Bewertungsprozess sowohl mit einer festen Musterlösung als auch mit beliebigen studentischen Referenzen genutzt werden.

7.c Ergebnisdarstellung

Wenn der Backend-Service antwortet, wird eine Ergebnissection eingeblendet:

- **Gesamtpunktzahl** (auf „0–100“ gerundet)
- **Pass/Fail-Status** (Schwellenwert 70 Punkte)
- **Detaillierte Einzelbewertungen** pro Bild (Kategorie, Punktzahl, Feedback, Stärken und Verbesserungen)
Alle Einträge werden in wiederverwendbaren „Result-Cards“ angezeigt, die mit Hilfe von DOM-Manipulation aus den JSON-Daten generiert werden.

