



School of
Management and Law

MLOps: Model Deployment & Maintenance

Vorlesung 4



Building Competence. Crossing Borders.

Adrian Moser, mosa@zhaw.ch, Institut für Wirtschaftsinformatik

Inhalte

ONNX

Struktur, Operatoren

Visualisierung

Runtime

ModelZoo

Konvertierung

Beispiel: efficientnet-lite4-onnx

ONNX

Open Neural Network Exchange

- definiert ein offenes Format zur Repräsentation von Deep-Learning-Modellen
- ONNX ist mit MIT-Lizenz über GitHub frei verfügbar

Entstehung

- ursprünglich gemeinschaftliches Projekt von Microsoft und Facebook
- mittlerweile unter dem Dach der «The Linux Foundation»



Vorteile

Freiere Wahl der Tools

- Frameworks wie Pytorch oder Keras sind für Training optimiert, aber nicht unbedingt für die Produktion
- Wahl des besten Frameworks für jeden Anwendungszweck

Performance

- Ausführung des Modells mit spezialisierten Frameworks, z.B. NVIDIA TensorRT

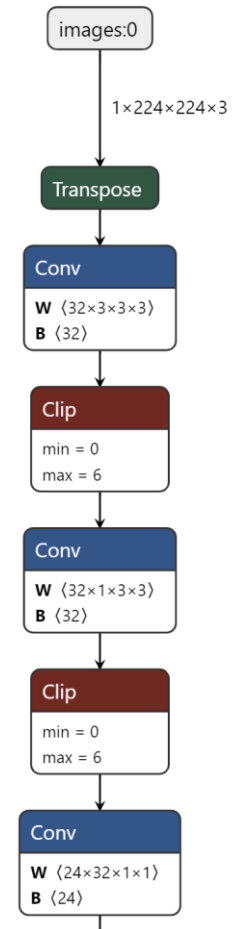
Kompatibilität

- keine aufwendige Migration oder Transformation der Modelle
- Proprietäre Modelle in Pytorch and Keras können bei Updates inkompatibel werden
- ONNX ist klar definiert und damit zukunftssicher

ONNX Interne Struktur

Graphstruktur

- ONNX beschreibt eine Graphstruktur
- Prozesse werden nacheinander auf den Eingabedaten ausgeführt



ONNX Operatoren

Operatoren

ONNX definiert eine Liste verfügbarer Operatoren

OpSet

Definiertes Set von Operatoren, welche in einer ONNX Datei verwendet werden können

Siehe

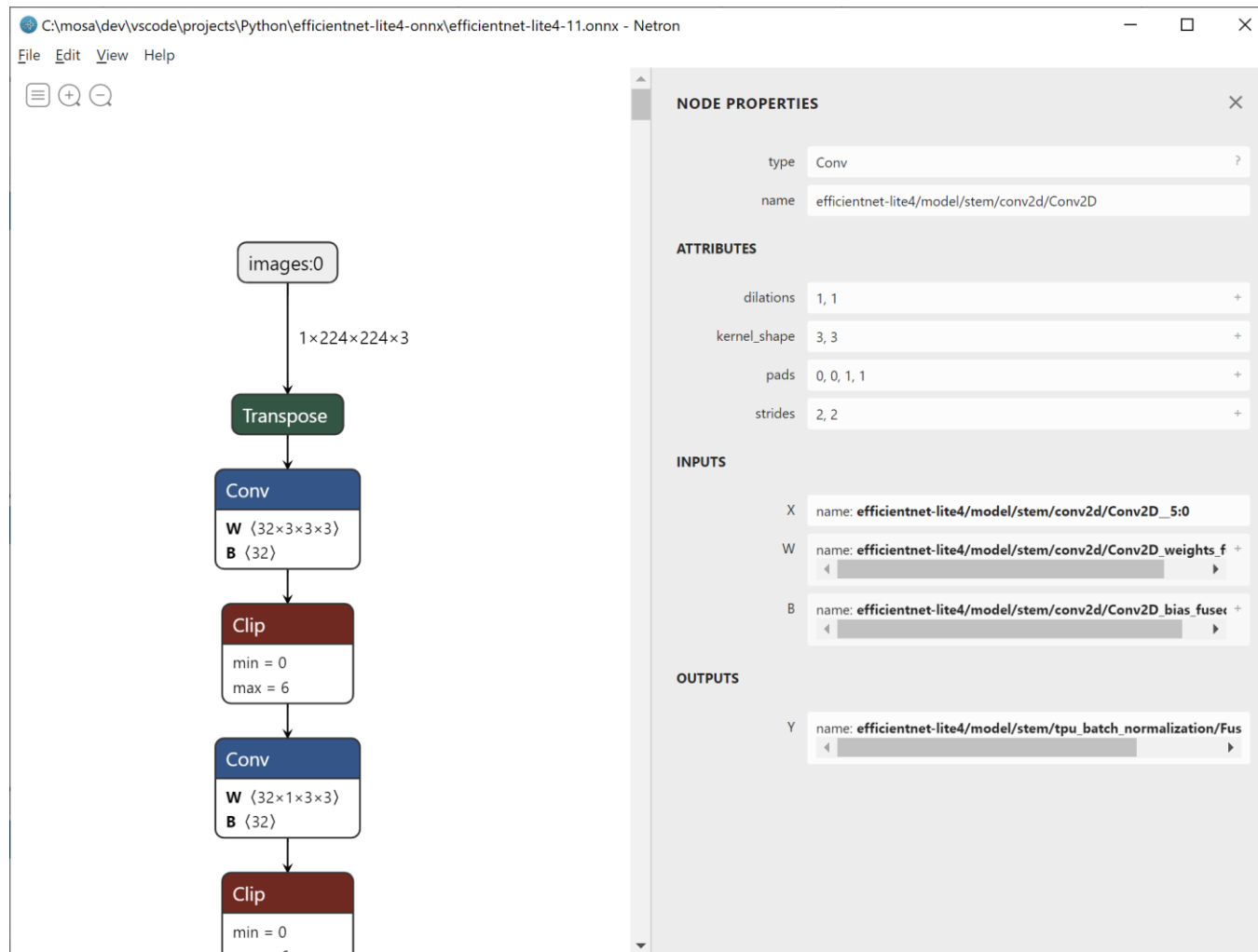
<https://github.com/onnx/onnx/blob/main/docs/Operators.md>

ai.onnx (default)

Operator	Since version
Abs	13, 6, 1
Acos	7
Acosh	9
Add	14, 13, 7, 6, 1
And	7, 1
ArgMax	13, 12, 11, 1
ArgMin	13, 12, 11, 1
Asin	7
Asinh	9
Atan	7
Atanh	9
AveragePool	11, 10, 7, 1
BatchNormalization	15, 14, 9, 7, 6, 1
BitShift	11
BitwiseAnd	18
BitwiseNot	18

ONNX Visualisierung

ONNX kann mit Netron visualisiert werden



ONNX Runtime

- Laden und Bereitstellung von ONNX Modellen
- Unterstützung diverser Umgebungen und Plattformen

Optimize Inferencing		Optimize Training												
Platform	Windows		Linux		Mac		Android		iOS		Web Browser (Preview)			
API	Python		C++		C#		C		Java		JS		Obj-C	WinRT
Architecture	X64			X86			ARM64			ARM32			IBM Power	
Hardware Acceleration	Default CPU			CoreML			CUDA			DirectML				
	NNAPI			oneDNN			OpenVINO			SNPE				
	TensorRT			ACL (Preview)			ArmNN (Preview)			CANN (Preview)				
	MIGraphX (Preview)			ROCm (Preview)			Rockchip NPU (Preview)			TVM (Preview)				
	Vitis AI (Preview)			XNNPACK (Preview)										
Installation Instructions	Please select a combination of resources													

ONNX Anwendungsfälle

Unterstützung unterschiedlicher Hardware/Betriebssysteme

Ein ML Modell in verschiedenen Umgebungen in Produktion bringen

Beispiel

- Entwicklung in Python, aber Deployment mit C#/C++/Java-App

ONNX Model Zoo

Sammlung von Modellen im ONNX Format

Inhalt

- Vortrainierte Modelle im ONNX Format
- Jupyter Notebook für Training und Ausführung des Modells

Verfügbar auf GitHub

- <https://github.com/onnx/models>

Konvertierung

Native Konvertierung

- PyTorch
- ... und weitere

ONNXMLTools

- Tensorflow/Keras
- scikit-learn
- Apple Core ML
- Spark ML (experimental)
- ... und weitere



Aufgabe

- Laden Sie ein ONNX Modell Ihrer Wahl aus dem Model Zoo
- Öffnen Sie das Modell mit dem Netron Model Viewer
- Analysieren Sie einige Operatoren
- Zeigen und diskutieren Sie das Resultat mit Ihrem Banknachbarn

ONNX-Beispiel: efficientnet-lite4-onnx

Modell

CNN / ConvNet (Convolutional Neural Network)

Entwickelt von Google

<https://github.com/onnx/models/tree/main/vision/classification/efficientnet-lite4>

Image Classification

Bild als Eingabe, als Resultat die Klassifizierung von Objekten auf dem Bild

Top-1 Score (Anteil korrekte Vorhersagen für erste Klasse): ca. 80%

Model	Download	Download (with sample test data)	ONNX version	Opset version	Top-1 accuracy (%)
EfficientNet-Lite4	51.9 MB	48.6 MB	1.7.0	11	80.4

ONNX-Beispiel: efficientnet-lite4-onnx

Datenset (Train and Validation)

Das Modell wurde mit folgenden Daten trainiert:

Bilder: COCO 2017 Training Images, Validation Images

Annotationen: Training/Validation Annotationen

<https://cocodataset.org/#download>

ONNX-Beispiel: efficientnet-lite4-onnx

Datenset online anschauen: <https://cocodataset.org/#explore>



efficientnet-lite4-onnx: Python

Neue Inference-Session erstellen

```
import onnxruntime  
  
...  
  
ort_session = onnxruntime.InferenceSession("efficientnet-lite4-11.onnx")
```

Das Modell befindet sich im Root des Projektes

efficientnet-lite4-onnx: Labels

Labels laden

```
# load the labels text file
labels = json.load(open("labels_map.txt", "r"))
```

labels_map.txt

```
{
  "0": "tench, Tinca tinca",
  "1": "goldfish, Carassius auratus",
  "2": "great white shark, white shark, man-eater, man-eating shark,
  "3": "tiger shark, Galeocerdo cuvieri",
  "4": "hammerhead, hammerhead shark",
  "5": "electric ray, crampfish, numbfish, torpedo",
  "6": "stingray",
  "7": "cock",
  "8": "hen",
  "9": "ostrich, Struthio camelus",
  ...
}
```

efficientnet-lite4-onnx: Inference

Run inference

Modell ausführen und Resultat formatiert zurückgeben

```
results = ort_session.run(["Softmax:0"], {"images:0": img_batch})[0]
result = reversed(results[0].argsort()[-5:])
resultStr = ""
first = True
for r in result:
    if first:
        resultStr = labels[str(r)] + " (" + str(results[0][r]) + ")"
        first = False
    else:
        resultStr = resultStr + "<br>" + labels[str(r)] + " (" + str(results[0][r]) + ")"

    print(r, labels[str(r)], results[0][r])

return resultStr
```

Softmax normalisiert den Vektor in eine Wahrscheinlichkeitsverteilung

App Service: Bereitstellungscenter

ZIP-Deploy nur für kleine Datenmengen sinnvoll, daher Bereitstellung über GitHub:

The screenshot shows the Microsoft Azure portal interface for the 'mosa-helloworld' App Service. The left sidebar contains a navigation menu with options like 'Übersicht', 'Aktivitätsprotokoll', 'Zugriffssteuerung (IAM)', 'Tags', 'Diagnose und Problembehandlung', 'Microsoft Defender for Cloud', 'Ereignisse (Vorschau)', 'Bereitstellung', 'Schnellstart', 'Bereitstellungsslots', and 'Bereitstellungscenter'. The main content area displays the 'Bereitstellungscenter' for 'mosa-helloworld'. The 'Einstellungen' (Settings) tab is active, showing a notification that the deployment is in production. Below this, there is a section for 'Quelle' (Source) with a dropdown menu labeled 'Codequelle auswählen'. A blue callout bubble points to this dropdown menu with the text 'Quelle: GitHub'. Another blue callout bubble points to the 'Bereitstellungscenter' link in the left sidebar with the text 'Bereitstellungscenter'.

App Service: Bulddienst

Quelle: **GitHub**, als Buildanbieter (vorerst) **App Service-Bulddienst** auswählen

Diagnose und Problembehandlung

Microsoft Defender for Cloud

Ereignisse (Vorschau)

Bereitstellung

Schnellstart

Bereitstellungsslots

Bereitstellungszentrum

Einstellungen

Konfiguration

Authentifizierung

Application Insights

Identität

Hiermit wird Code von Ihrem bevorzugten Quell- und Buildanbieter bereitgestellt und erstellt. [Weitere Informationen](#)

Quelle *

GitHub

Erstellung mit GitHub Actions. [Anbieter ändern](#)

GitHub

App Service platziert einen GitHub Actions-Buildanbieter, um Ihre App zu erstellen und bereitstellen. Wenn Sie eine andere Buildanbieter auswählen müssen Sie möglicherweise zusätzliche Konfigurationen vornehmen. [Informationen](#)

Angemeldet als

innovad [Konto ändern](#) ⓘ

Buildanbieter

☐ GitHub Actions

☒ App Service-Bulddienst

☐ Azure Pipelines

OK **Abbrechen**

Repository, um ein Commit zu erstellen, wenn Sie es wünschen. [Weitere Informationen](#)

App Service: Code-Quelle

Code-Quelle auswählen:

Hiermit wird Code von Ihrem bevorzugten Quell- und Buildanbieter bereitgestellt und erstellt. [Weitere Informationen](#)

Quelle *

GitHub



Erstellung mit App Service-Builddienst. [Anbieter ändern](#)

GitHub

App Service platziert einen Webhook im ausgewählten Repository. Wenn ein neuer Commit per Push an die ausgewählte Verzweigung übertragen wird, ruft App Service Ihren Code ab, erstellt Ihre Anwendung und stellt sie in Ihrer Web-App bereit. Wenn Sie Ihre Organisation oder Ihr Repository nicht finden können, müssen Sie möglicherweise zusätzliche Berechtigungen für GitHub aktivieren. [Weitere Informationen](#)

Angemeldet als

innovad [Konto ändern](#) ⓘ

Organisation *

zhaw-iwi



Repository *

MLOps-efficientnet-lite4-onnx



Branch *

main

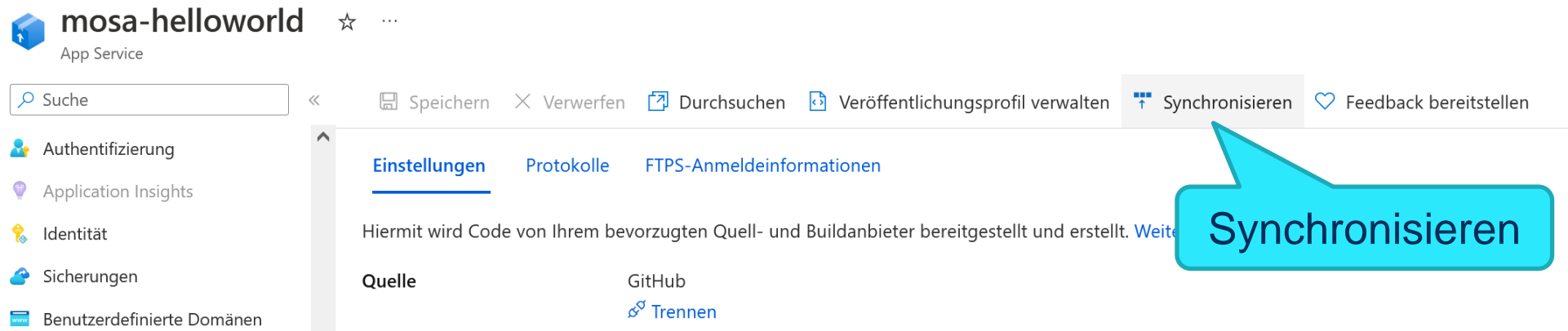


GitHub
Repository

App Service: Synchronisieren

GitHub Repository mit App Service synchronisieren

Es wird ein Build gestartet (bei Bedarf Log (Protokollstream) anschauen)



Deployment

ML EfficientNet-Lite4

ONNX Model Demonstration by mosa (Version 1.0)


Bitte Bild hochladen (jpeg, png)

train.jpg

Answer:

passenger car, coach, carriage (0.4928616)
streetcar, tram, tramcar, trolley, trolley car (0.34702685)
electric locomotive (0.1560171)
trolleybus, trolley coach, trackless trolley (0.0031235914)
freight car (0.00012333854)

Analyzed Image:



Links:
[EfficientNet-Lite4 Repository](#)

Aufgabe

- Wählen Sie ein Modell aus dem ONNX Model Zoo (Dateigrösse beachten!)
- Integrieren Sie das Modell in Python
- Erstellen Sie ein einfaches User Interface
- Deployen Sie die App auf dem Azure App Service mittels App Service Builddienst