Vergleich von Optimizern mit variablen Lernraten

Die Effizienz und Konvergenzgeschwindigkeit eines neuronalen Netzwerks hängen stark von der Wahl des Optimierers und der Lernrate ab. Um die Auswirkungen verschiedener Optimierer und Lernraten auf die Trainingsleistung zu analysieren, wurde dieses Experiment entworfen. Drei verschiedene Optimierer - Stochastic Gradient Descent (SGD), Adam und RMSprop - werden im Hinblick auf ihre Fähigkeit verglichen, ein neuronales Netzwerk auf dem MNIST-Datensatz zu trainieren.

Datensatzvorbereitung:

- MNIST steht für "Modified National Institute of Standards and Technology"
- Der MNIST-Datensatz ist ein umfangreicher und weit verbreiteter Datensatz in der maschinellen Lerngemeinschaft
- Der Datensatz besteht aus handgeschriebenen Ziffern (0 bis 9), die in Graustufen auf 28x28 Pixel großen Bildern abgebildet sind.
- Die Pixelwerte liegen im Bereich von 0 bis 255, wobei 0 Weiß repräsentiert und 255 Schwarz
- Es gibt insgesamt 60.000 Trainingsbilder und 10.000 Testbilder
- Das Hauptziel bei der Verwendung des MNIST-Datensatzes besteht darin, Modelle zu erstellen, die in der Lage sind, die handgeschriebenen Ziffern korrekt zu klassifizieren

Gewählte Optimierer

- Stochastic Gradient Descent (SGD):
  SGD ist der grundlegende Optimierungsalgorithmus. Er aktualisiert die Modellgewichte basierend auf dem Gradienten der Verlustfunktion für jede Trainingsinstanz einzeln. Es erfordert möglicherweise die sorgfältige Einstellung der Lernrate. Es kann zu langsamem Konvergieren und Schwierigkeiten bei lokalen Minima führen.
- Adam (Adaptive Moment Estimation):
  Adam ist ein adaptiver Optimierer, der eine Kombination aus Momenten des Gradienten (ähnlich wie beim SGD) und einer Schätzung der unzentrierten Varianz verwendet. Dies ermöglicht eine adaptive Anpassung der Lernrate für jeden Parameter. Es werden weniger manuelle Abstimmung der Hyperparameter erfordert im Vergleich zu SGD.
- RMSprop (Root Mean Square Propagation):
  RMSprop ist ein adaptiver Optimierer, der die Lernrate für jeden Parameter basierend auf einem gleitenden Durchschnitt der Quadratwerte der vorherigen Gradienten anpasst. Dies hilft, die Lernrate dynamisch zu steuern.
  RMSprop adressiert einige Herausforderungen von SGD, wie das Anpassen der Lernrate für verschiedene Parameter.

Informationen zu den Optimierern: https://keras.io/api/optimizers

Neuronales Netzwerk:

```
marvin = tf.keras.models.Sequential([
  tf.keras.layers.InputLayer(input_shape=(28,28,1)),
  tf.keras.layers.Conv2D(filters=32, kernel_size=5),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

- Input Layer:
  Die Input Layer definiert die Form der Eingabedaten. In diesem Fall haben die Eingabebilder eine Größe von (28, 28, 1), was auf 28x28 Pixel-Bilder mit einem einzelnen Kanal (z. B. Graustufenbilder) hinweist.
- Convolutional Layer (Conv2D):
  Dieser Layer führt eine 2D-Faltung auf den Eingabebildern durch. Es werden 32 Filter mit einer Kernelgröße von 5x5 verwendet, um Merkmale aus den Bildern zu extrahieren. Die Aktivierungsfunktion ist standardmäßig linear, was bedeutet, dass keine nicht-linearäre Aktivierung nach der Faltung durchgeführt wird.
- Flatten Layer:
  Dieser Layer wird verwendet, um die Ausgabe des vorherigen Convolutional Layers zu flatten. Er konvertiert die mehrdimensionalen Daten in einen flachen Vektor, der als Eingabe für die Dense-Schicht dienen kann.
- Dense Layer:
  Ein Dense Layer mit 64 Neuronen und ReLU-Aktivierungsfunktion. Diese Schicht führt eine vollständig verbundene Operation auf den flachen Daten aus, wodurch komplexe nicht-lineare Transformationen ermöglicht werden.
- Dropout Layer:
  Dropout wird verwendet, um Overfitting zu reduzieren. Der Dropout-Layer deaktiviert zufällig einen bestimmten Prozentsatz der Neuronen während des Trainings, um die Robustheit des Modells zu verbessern.
- Dense Layer (Output Layer):
  Die Ausgabeschicht besteht aus 10 Neuronen, die mit der Softmax-Aktivierungsfunktion aktiviert sind. Dies ist typisch für Klassifikationsprobleme mit mehreren Klassen (in diesem Fall 10 Klassen für die Ziffern 0-9). Die Softmax-Aktivierung gibt Wahrscheinlichkeiten für jede Klasse aus

Variable Lernraten:

- Jeder Optimierer wird mit drei verschiedenen Lernraten trainiert, um ihre Sensitivität gegenüber unterschiedlichen Lernraten zu beurteilen.
- Hierzu wird jeweils eine große/mittlere/kleine Lernrate gewählt

Trainingsablauf:

- Das Training erfolgt über 100 Epochen, wobei es vorzeitig abgebrochen wird, wenn nach 5 aufeinanderfolgenden Epochen keine Verbesserung der Validierungsgenauigkeit festgestellt wird
- Die Batch-Size wird auf 128 festgelegt

Ziel und Hypothese:

- Das Ziel dieses Experiments ist es, herauszufinden, wie sich die Kombination von Optimierern und Lernraten auf die Trainingsleistung von neuronalen Netzwerken auswirkt. Wir erwarten, dass verschiedene Optimierer bei unterschiedlichen Lernraten unterschiedlich gut abschneiden, und wir möchten feststellen, welche Kombination die besten Ergebnisse auf dem MNIST-Datensatz liefert

Ergebnisse

Optimizer: Adam

Learning Rate	Epochen	loss	accuracy	val_loss	val_accuracy
0.003	7	0.0757	0.9762	0.1682	0.9684
0.001	9	0.0346	0.9885	0.1459	0.9718
0.0001	12	0.0303	0.9898	0.1143	0.9758

Optimizer: SGD

Learning Rate	Epochen	loss	accuracy	val_loss	val_accuracy
0.003	6	0.0366	0.9874	0.1515	0.9720
0.001	7	0.0359	0.9883	0.1495	0.9717
0.0001	11	0.0367	0.9878	0.1493	0.9722

Optimizer: RMSprop

Learning Rate	Epochen	loss	accuracy	val_loss	val_accuracy
0.003	8	0.0842	0.9807	0.3330	0.9617
0.001	15	0.0319	0.9905	0.1996	0.9704
0.0001	26	0.0293	0.9905	0.1561	0.9736

Die erzielten Ergebnisse zeigen, dass bei Verwendung einer kleineren Lernrate alle Optimierer dazu neigen, höhere Epochenanzahlen zu benötigen, um eine konvergente Genauigkeit zu erreichen. Dieser Zusammenhang ist darauf zurückzuführen, dass eine kleinere Lernrate den Aktualisierungsprozess der Modellgewichte verlangsamt, was zu einem vorsichtigeren und schrittweisen Lernprozess führt. In diesem Szenario haben die Optimierer mehr Zeit benötigt, um die relevanten Merkmale der Trainingsdaten zu erfassen und die Modellleistung zu verbessern. Die steigende Genauigkeit über die zusätzlichen Epochen deutet darauf hin, dass trotz der längeren Trainingsdauer die Modelle mit kleinerer Lernrate eine konstante Verbesserung erfahren und allmählich konvergieren. Dieser Befund unterstreicht die Bedeutung der Lernratenwahl bei der Anpassung von Optimierern, da zu kleine Lernraten zu längeren Trainingszeiten führen können, aber gleichzeitig eine stabilere Konvergenz ermöglichen.