AI Projektdokumentation

Autoencoder

Documentation

by

Alexander Grissinger

Matriculation number: 11006444

2019-07-21

SRH University Heidelberg

School of Information, Media and Design

Degree course "Applied Artificial Intelligence"

Major field "Applied Computer Science (M.Sc.)"

Reviewers Dr. Simon Ziegler

Table of Contents

1 Übersicht

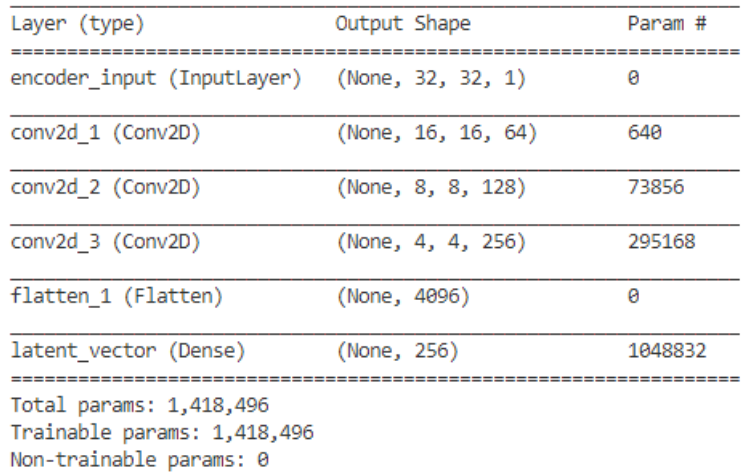
Ziel des Projekts ist die Implementierung eines Autoencoders, welcher Bildern, die einen RGB Farbwert besitzen, aber grau skaliert wurden, wieder einen RGB Wert zuordnen kann. Für die Implementierung wurde die Skriptsprache Python benutzt. Ebenso wurde Keras für dieses Projekt importiert und als Backend fundiert TensorFlow. Als Datenset wird das Cifar10 Datenset verwendet, welches eine große Bandbreite an Bildern aus verschiedenen Kategorien anbietet.

2 Umsetzung

Zur Einarbeitung bezüglich Autoencoders, CNN und der Nutzung von Keras wurde die Website https://blog.keras.io/building-autoencoders-in-keras.html genutzt. Wie die Implementierung des Cifar10 Datensets funktioniert wurde anhand des Codebeispiels https://keras.io/examples/cifar10_cnn/ gelernt. Zur weiteren Recherche wurde das Buch „Advanced Deep Learning with Keras“ genutzt.

Zur Veranschaulichung des Beispiels werden die ersten 100 Vorkommen im Cifar10 Datenset geplotted. Diese werden ebenso auf die Größe 10x10 Pixel skaliert.

Abbildung 1: Erste 100 Vorkommen im Cifar 10 Datenset

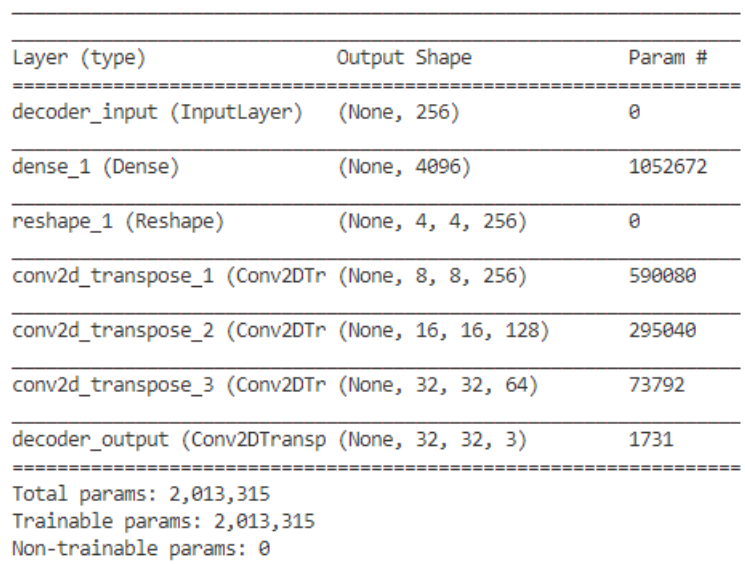


Layer (type)	Output Shape	Param #
encoder_input (InputLayer)	(None, 32, 32, 1)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	640
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_3 (Conv2D)	(None, 4, 4, 256)	295168
flatten_1 (Flatten)	(None, 4096)	0
latent_vector (Dense)	(None, 256)	1048832

Total params: 1,418,496
Trainable params: 1,418,496
Non-trainable params: 0

Abbildung 2: Ergebnis der Enkodierung

Im obigen Bild sieht man das Ergebnis nach der Enkodierung des Datensets. Die 32 steht in diesem Fall für die Anzahl von Zeilen und Spalten.



Layer (type)	Output Shape	Param #
decoder_input (InputLayer)	(None, 256)	0
dense_1 (Dense)	(None, 4096)	1052672
reshape_1 (Reshape)	(None, 4, 4, 256)	0
conv2d_transpose_1 (Conv2DTr	(None, 8, 8, 256)	590080
conv2d_transpose_2 (Conv2DTr	(None, 16, 16, 128)	295040
conv2d_transpose_3 (Conv2DTr	(None, 32, 32, 64)	73792
decoder_output (Conv2DTransp	(None, 32, 32, 3)	1731

Total params: 2,013,315
Trainable params: 2,013,315
Non-trainable params: 0

Abbildung 3: Result Decoder

Die Bilder in Abbildung 1 werden im folgenden Schritt grau skaliert. Das Ergebnis kann im nächsten Bild betrachtet werden.

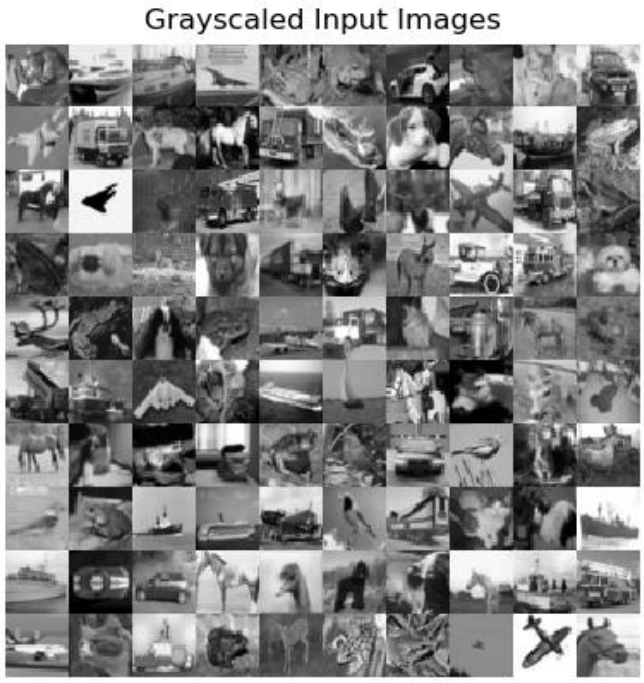


Abbildung 4: Grauskalierte Bilder

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   (None, 32, 32, 1)         0
_________________________________________________________________
encoder (Model)              (None, 256)               1418496
_________________________________________________________________
decoder (Model)              (None, 32, 32, 3)         2013315
=================================================================
Total params: 3,431,811
Trainable params: 3,431,811
Non-trainable params: 0
```

Abbildung 5: Result Autoencoder vor Trainingsstart

Nun wird der Autoencoder mit einem Traingset trainiert und anschließend getestet. Dies geschieht innerhalb von 30 Epochen. Nach diesem Training soll der Autoencoder den grau

skalierten Bildern wieder die originalen Farbwerte zuordnen. Das Ergebnis lässt sich im folgenden Bild ansehen.

Bei Nutzung einer Nvidia Geforce GTX 950M dauert das Training von 50000 Samples pro Epoche etwa 80-90 Sekunden, wobei 90 Sekunden eher die Ausnahme bilden und die meisten Epochen um die 80 Sekunden benötigen. Laut CUDA gehört diese Grafikkarte eher zu den mittel-bis unterklassigen Grafikkarten, somit ist bei neueren bzw. Karten mit besserer Rechenkapazität auch ein schnelleres Training denkbar. Zu sehen im folgenden Bild.

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
2019-07-21 22:23:33.516017: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports ins
2019-07-21 22:23:33.523936: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successful
2019-07-21 22:23:34.084368: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1640] Found device 0 wi
name: GeForce GTX 950M major: 5 minor: 0 memoryClockRate(GHz): 0.928
pciBusID: 0000:01:00.0
2019-07-21 22:23:34.090168: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] G
2019-07-21 22:23:34.095167: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1763] Adding visible gp
2019-07-21 22:23:38.926406: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1181] Device interconne
2019-07-21 22:23:38.929938: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1187]      0
2019-07-21 22:23:38.931963: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1200] 0:   N
2019-07-21 22:23:38.937349: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlo
50M, pci bus id: 0000:01:00.0, compute capability: 5.0)
50000/50000 [==============================] - 90s 2ms/step - loss: 0.0155 - val_loss: 0.0117
Epoch 2/30
50000/50000 [==============================] - 80s 2ms/step - loss: 0.0102 - val_loss: 0.0096
Epoch 3/30
50000/50000 [==============================] - 80s 2ms/step - loss: 0.0092 - val_loss: 0.0088
Epoch 4/30
46496/50000 [==========================>...] - ETA: 5s - loss: 0.0086
```

Abbildung 6: Auszug aus Training des Datensets

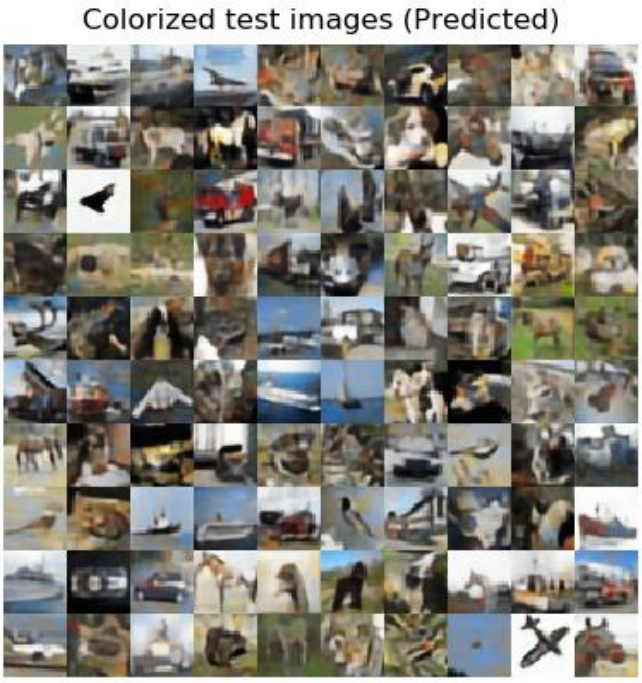


Abbildung 7: Output Autoencoder

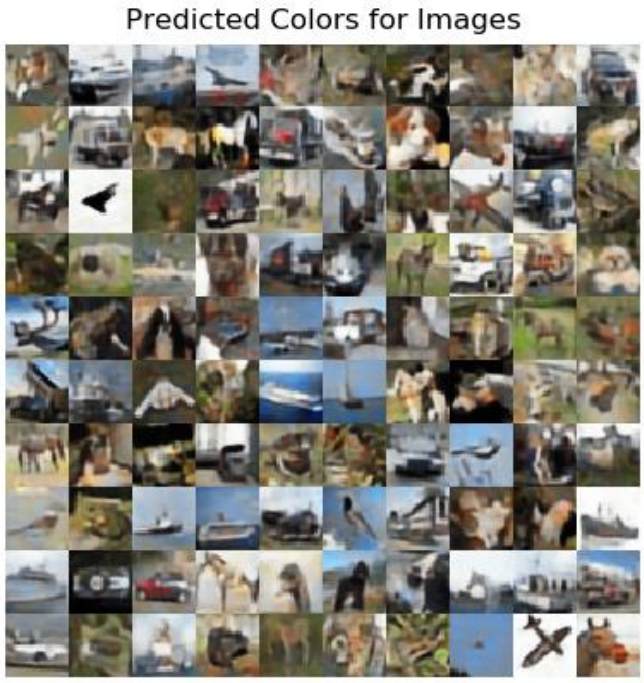


Abbildung 8: Output 2 Autoencoder

Es ist deutlich zu erkennen, dass der Autoencoder im Vergleich zum Originalbild einige Schwächen bezüglich der Sättigung aufweist, ebenso gibt es einige Probleme bei der Darstellung roter Gegenstände. Dennoch ist das Ergebnis zufriedenstellend, da Farbwerte neu zugeordnet werden konnten und die Objekte innerhalb der einzelnen Bilder gut zu erkennen sind.