Colorization of black-and-white images with Deep Learning

Studienprojekt – Daniel Ketterer

Colorization

- Schwarz-weiß Bilder farbig machen
- Entfernen von Farbe aus einem Bild ist surjektiv
- Es gibt mehrere plausible farbige Lösungen zu einem SW-Bild

Quelle: http://arxiv.org/abs/1705.07208, rechts Original

Agenda

- Scribble-based Colorization
- Example-based Colorization
- Automatic Colorization
- Colorization mit U-Net und Regression Loss

Scribble-based Colorization

- Nutzen Farb-Hinweise als Striche oder Punkte die von einer Nutzerin erzeugt werden
- Funktioniert ohne Deep-Learning
- Idee z.B.:
  - Nachbarpixel mit derselben Lichtintensität haben dieselbe Farbe
  - Ähnliche Texturen haben dieselbe Farbe
  - → Hinweise über das Bild verteilen
  - Segmentation des Bildes um Colorbleeding zu verhindern
- Scribbles brauchen Übung und Erfahrung im Zusammenspiel mit den Algorithmen

Quelle: Levin et al., 2004, Colorization Using Optimization

Example-based Colorization

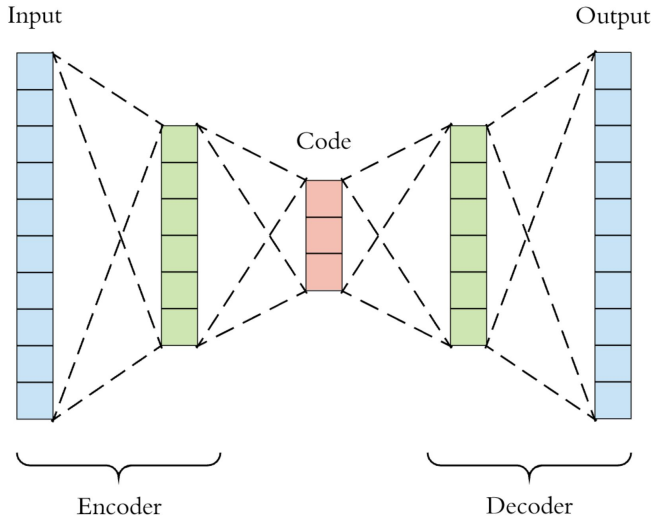
- SW-Bild wie Referenz Bild kolorieren
- Statistische Analysen und Histogramm Abstimmung
- Deep Convolutional Neural Networks (DCNN) mit der Referenz und dem SW-Bild trainieren
- Aligned Referenz R’ und Ähnlichkeits-Matching mit Deep Image Analogy (Liao et al., 2017)
- Vorverarbeitung mit VGG-19 für Ähnlichkeits-Matching
- Autoencoder-DCNN für die Farbvorhersage

Target	Reference	Aligned reference	Predicted result	Chrominance difference	Matching error	Matching error
T_L	R	R'	$T_L \bigoplus R'_{ab}$	$\lvert P_{ab} - R'_{ab} \rvert$	$1 - sim_{T \to R}$	$1 - sim_{R \to T}$

Quelle: http://arxiv.org/abs/1807.06587

Automatic Colorization

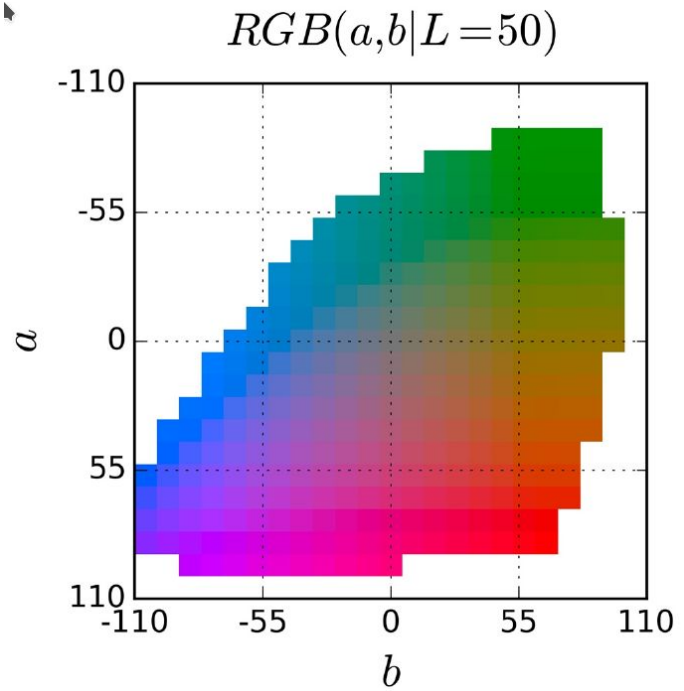
- DCNN das ohne weitere Hilfe zu einem SW-Bild plausible Farben findet
- Verwenden alle irgendeine Autoencoder Architektur
- Ausgabe als:
  - direkte Vorhersage der Farbwerte
  - Verteilung, aus der die Farbwerte abgeleitet werden
- Loss als:
  - L1/L2-Regression loss
  - cGAN
  - Cross-entropy / KL-Divergenz



Quelle:
https://miro.medium.com/max/1762/1*oLIbsQnYKX5DEnMQK3pH_Ig.png

Colorization als Klassifikation

- Unterteilung des (2D-) Farbraums in Klassen
- Training mit Cross-entropy
- Höhere Gewichtung seltener Klassen während des Trainings
- Prediction mit:
  - dem wahrscheinlichstem Wert
  - annealed-mean



Colorization als Regression

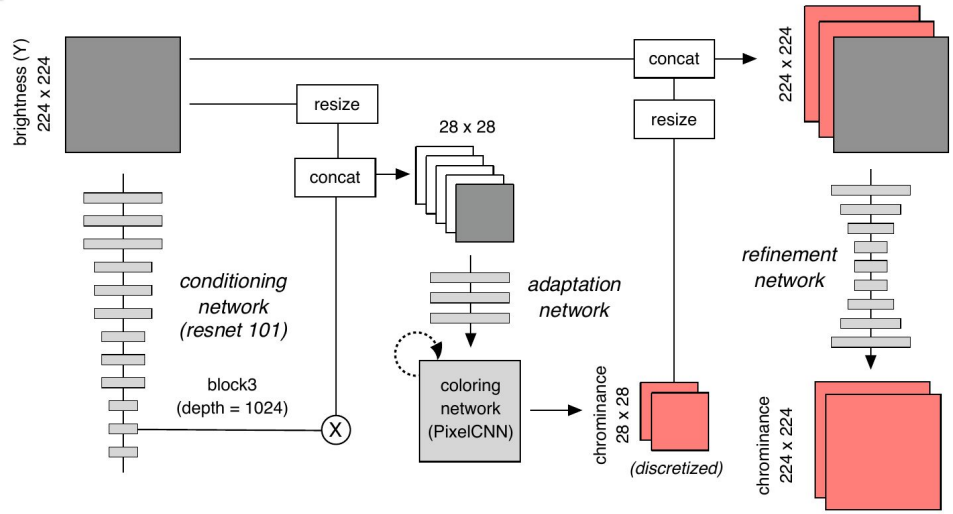
- Direkte Vorhersage von Werten im entsprechenden Farbraum
- Kombiniertes Training mit Klassifikation von Bildern möglich
- Vorteil, da bekannt das Training für Klassifikation zuverlässig gute Filter ergibt
- Autoencoder (Hypercolumn) Architektur
- L1- / L2-Norm als Loss-Funktion
- Eher desaturierte Ergebnisse, Objekte oft nur zum Teil gefärbt, “Color-Bleeding”

Colorization mit Conditional GAN

- Generator produziert direkte Farbwerte
- Generator hat Zufallsvektor und SW-Bild als Eingabe
- Diskriminator muss Ausgabe von Generator und echte Farben unterscheiden
- Diskriminator bekommt auch das SW-Bild als Eingabe
- Training mit L1 + cGAN Loss für die besten Ergebnisse
- Training als PatchGAN mit 70x70 Pixel großen Patches

PixColor: PixelCNN + Refinement CNN

- PixelCNN:
  - Hat zufälligen Startwert
  - Nimmt Global Features und SW-Bild
  - Gibt 2 Farbkanäle (U + V) 28x28
- Refinement CNN:
  - Nimmt vergrößerte Farbkanäle + SW-Bild
  - Gibt 2 Farbkanäle in Originalgröße
- Unabhängiges Training von PixelCNN und Refinement CNN



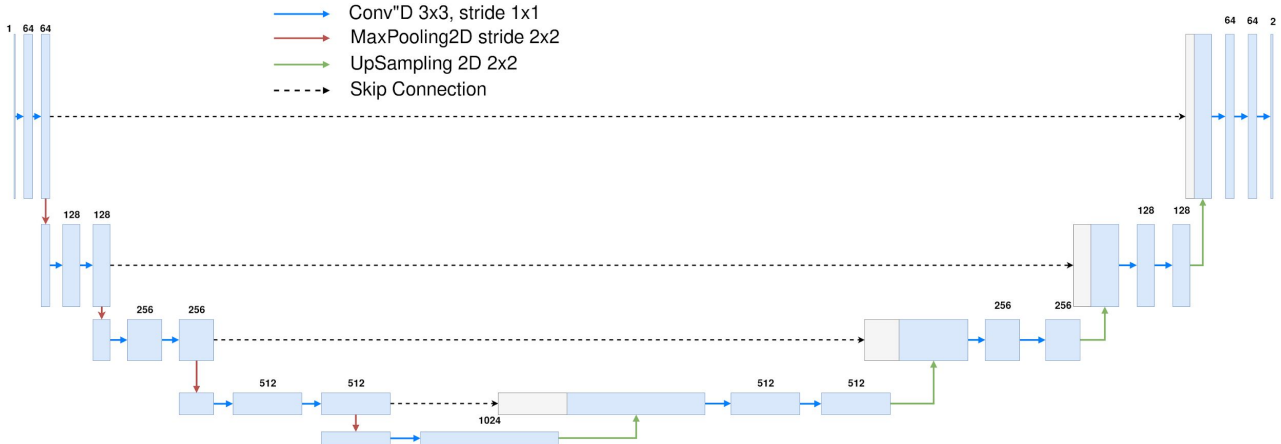
Eigene Arbeit

Training und Evaluation eines
Colorization Netzes mit Keras + TF

```
model.fit_generator(generator.generate(batch_size),
                    epochs=args.epochs,
                    initial_epoch=init_epochs,
                    validation_data=val_generator.generate(batch_size),
                    steps_per_epoch=generator.size // batch_size,
                    validation_steps=val_generator.size // batch_size,
                    callbacks=[tensorboard, lrtensorboard, ckptSaver])
```

U-Net und Regression Loss

- U-Net: Autoencoder mit Skip-Connections
- CIE Lab Farbraum
- Vorhersage der ab Kanäle
- Training mit L1- und L2-Norm
- Conv Aktivierung: ReLU
- Letztes Layer: tanh



Datensatz

- ImageNet ILSVRC 2012
  - ~1.2M Bilder im Trainingsset
  - 50000 Bilder im Validierungsset
  - 1000 unterschiedliche Kategorien
  - Objekte in natürlichem Kontext

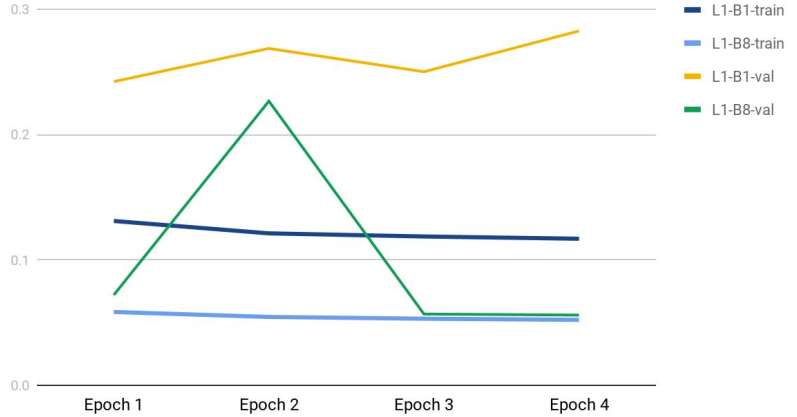
Training

- Aufteilung in L und ab der Bilder
- Verschiebung in des Intervall [-1, 1]
- Größe Anpassen: längste Seite 256 Pixel
- Zero-Padding zu 256x256 → Gleiche Größen im Batch
- ADAM mit Lernrate: 0.0003 als Optimierer
- 4 Epochen Training
- Kein Transferlernen

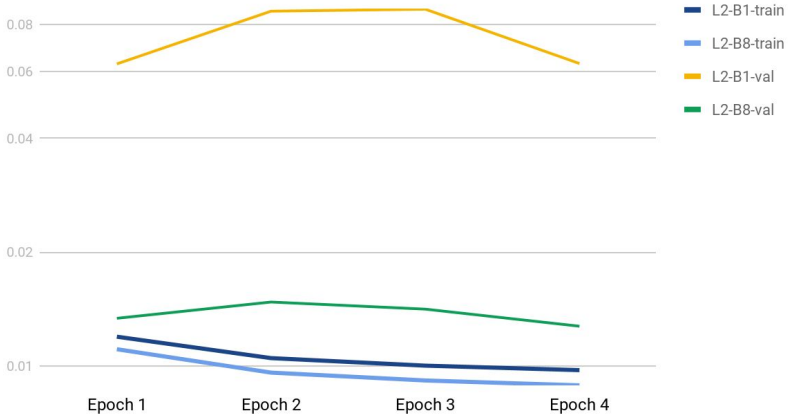
→ Starkes Overfitting bei Batch size 1

→ Moderates Overfitting bei Batch size 8

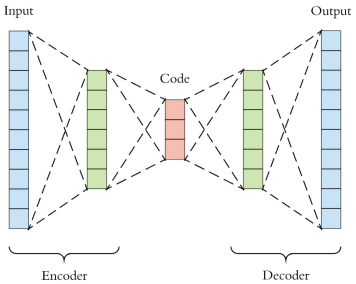
Loss beim Training mit L1



Loss beim Training mit L2



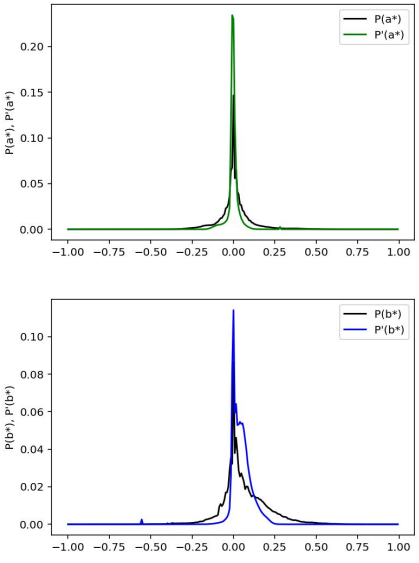
Ergebnisse



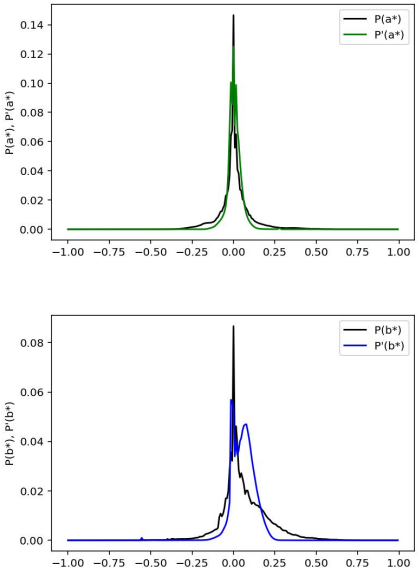
Beispiele

Histogram Intersection

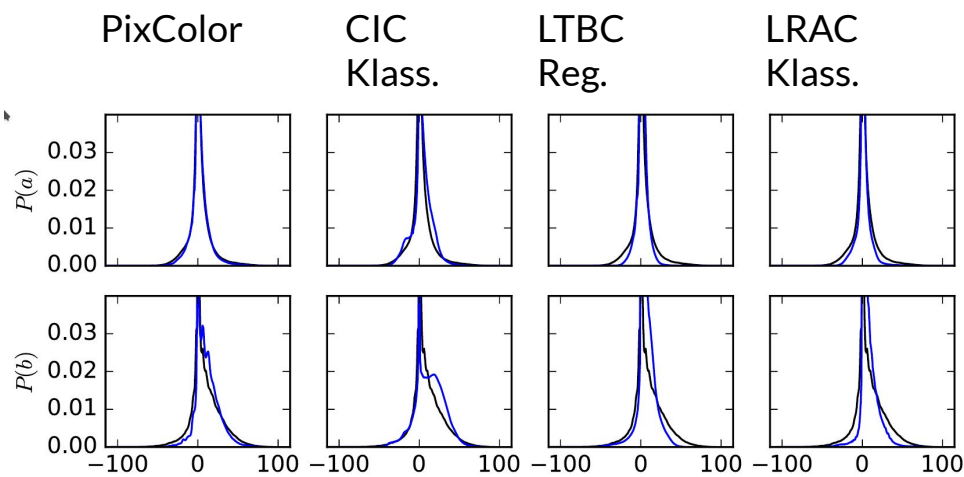
	U-Net L1	U-Net L2	CIC	LTBC	LRAC	PixColor	L1 + cGAN
a	0.631	0.768	0.85	0.82	0.78	0.93	0.84
b	0.648	0.683	0.85	0.82	0.78	0.93	0.82



U-Net L1

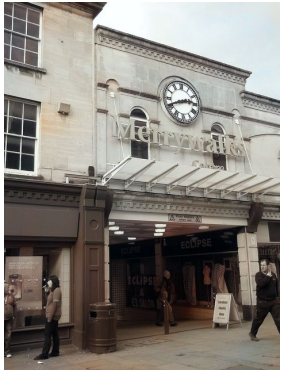


U-Net L2



Schwarz: Verteilung ImageNet val1k
Grün: a* Blau: b*

Typische Fehler



Desaturiert

Inkonsistente Farben

Kaputte Ecken

Farben verlaufen

Fazit

- Ergebnisse mit Regression sind visuell nicht zufriedenstellend
- Histogram-Instersection in der Nähe der Literaturwerte
- Mögliche Verbesserungen durch:
  - Transferlernen
  - Normalisierung der Daten
  - Finetuning der Lernrate
  - Größere Batchsize auf anderer GPU
- Thematisch große Schnittmengen mit spannenden Themen aus dem DL: cGAN, PixelCNN

Vielen Dank für die Aufmerksamkeit!