Fragen:

- Wie sehr müssen wir uns an dem Code in den Folien orientieren und wie viel Freiheit haben wir neue Anwendungen zu erstellen
- Ideen: style transfer von Kunstepochen zu Bildern, increase image resolution, turning daytime scene into nighttime ; Synthesizing satellite view from map view
- Wie können wir unseren code per email verschicken / im ilearn hochladen
- Werden wir zusammen bewertet?

- Eigener Beitrag
- Aufgabenstellung
- Timing
- Leutemitnehmen
- Stoff den Leuten beibringen
- In Häppchen unterteilen
- Gute Folien
- Passender Inhalt

Cuda:

C:\Users\Raoul\AppData\Local\Temp\cuda

Todo:

- Git push!
- trainieren
- Auswendig lernen
- Mehr Visualisierungen im code
- Show different model architecture performances
- Clean up and provide repo
- Cgan auf andere Anwendungen

Ideen:

- Kann ich bilder in google colab einfügen:
-

Script

Möglichkeiten von CGANS:

- Foto vom Tag zu Nacht
- Synthethisiere sattelitenbilder von Landkartenbildern

Image-to-Image Translation (Pix2Pix)

Super-Resolution (SRGAN)

Data Augmentation

Style Transfer

-

Live Coding:

```
def build_generator(latent_dim, num_classes=10):
  # Latent input and label
  latent = Input(shape=(latent_dim,))
  image_class = Input(shape=(1,), dtype='int32')

  # Embedding for categorical input
  cls = Embedding(num_classes, latent_dim)(image_class)
  cls = Flatten()(cls)
  model_input = Multiply()([latent, cls])
```

```
def build_discriminator(num_classes=10):
  img = Input(shape=(28, 28))
  img_class = Input(shape=(1,), dtype='int32')

  # Embedding for categorical input
```

```
cls = Embedding(num_classes, 784)(img_class)
cls = Flatten()(cls)
flat_img = Flatten()(img)

# Merged model by multiplying
model_input = Multiply()([flat_img, cls])
```

- Latent Space: hier wird die Zuordnung von Labels und Gelernten Bildern erstellt ; bedeutung wird encoded
- Beta: decay rate vom moving average des Gradienten ; niedrigere Werte legen mehr wert auf den derzeitigen Gradienten
- Embedding layer : turns input labels into dense vectors of noise vector shape
- BatchNormalization(): normalizes the inputs, in order to reduce the variance during gradient descent ; calculates the mean and the variance of the activations ; produces outputs close to 0 and a standard deviation close to 1
- Momentum: wie sehr wird durch die Normalisierung die Varianz des Batches abgeflacht ; 1 ist niedrige fluktuation ; **hohes Momentum macht das model stabiler, niedriges momentum lässt das model mehr durch neue Daten beeinflussen**

Wichtig:

- Falls jemand fragen hat gerne stellen
- Training results interpretieren
- Frage stellen, warum training suboptimal ist

CGAN Discriminator:

1. Label gets transformed into dense vector by embedding layer
2. Reshaping
3. Concatenate reshaped label embedding onto corresponding image (stamp it on top of it)
4. Feed the joint representation as input into the CGAN Discriminator network

- Model input dimensions have to be adjusted to size x*x*2 because of the new input shape due to stamping

- Dept of first convolutional layer is doubled from 32 to 64 due to bigger image size
- Output layer: sigmoid for evaluation

Building the model:

- Same input layer is passer to the generator and the Discriminator compared to other GANs

Def sample_images():

- Used to examine how the quality of generated images improves as training progresses
- We create 2 grids of numbers ; 1-5 and 6-9
- Alloys us to inspect how well specific numerals are produced