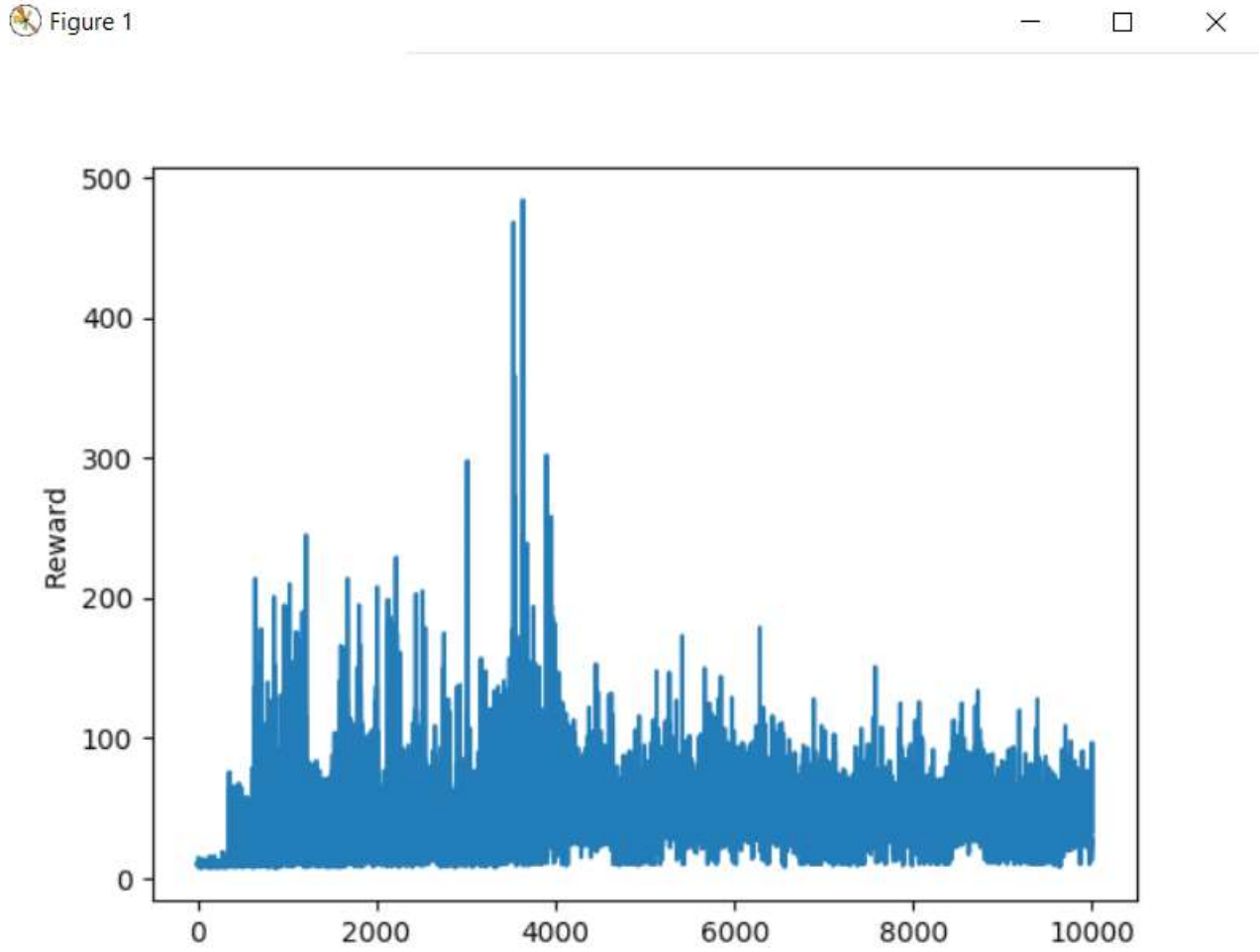
Reinforcement learning

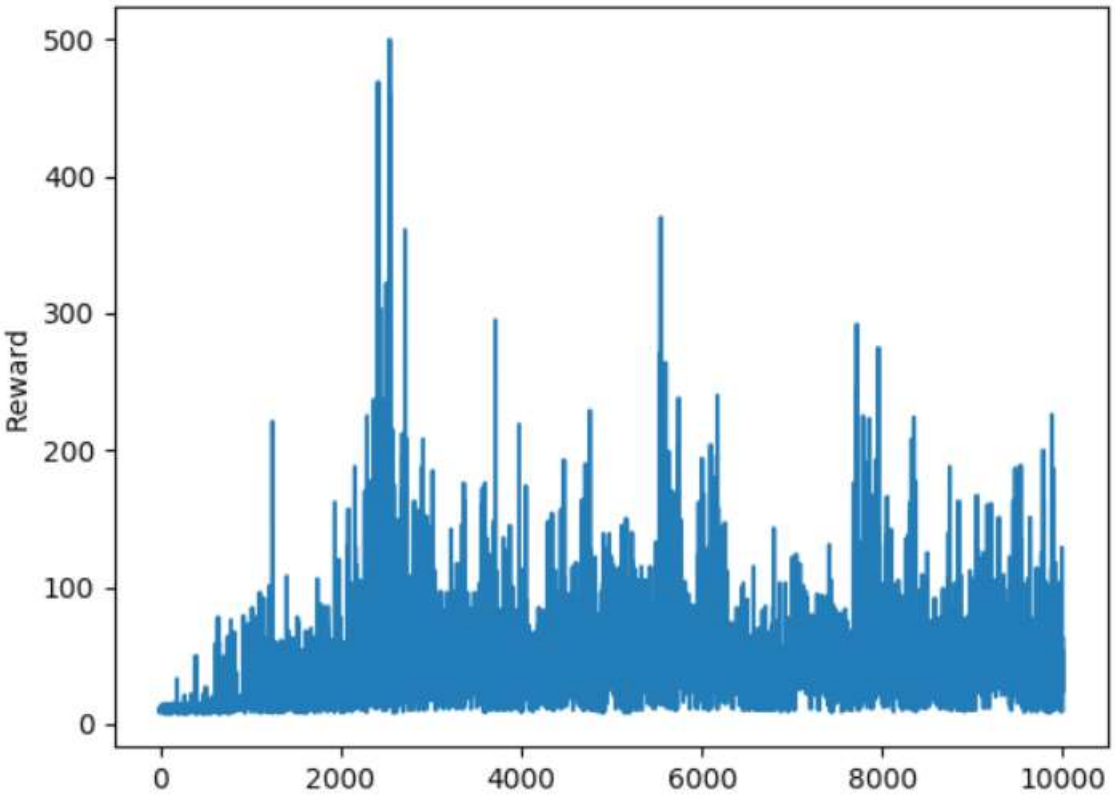
Source: reinforcement_learning_pole_cart_gym.py

De fyra mätningarna som ingår i state-informationen är: ”cart position”, ”cart velocity”, ”pole angle”, ”pole velocity at tip”. Det första metoden gör är att initierar Q-matrisen med nollor, sedan skapas ett Q-matris-index från de fyra mätvärderna. Indexet används för i Q-matrisen för att slå upp ett av två värden som är högst, ger högst reward, antingen action 0 eller 1 vilket betyder dra cart åt vänster eller höger. Metoden kan också välja att slumpmässigt ta en action med 10% sannolikhet. Actionen utförs och om den misslyckades startas en ny episod, innan dess uppdateras rewarden för det state som just utfördes.

Nedan är en bild av den kumulativa rewarden, med EPISLON=0,2, GAMMA=0,9, från 0 till 10000 episoder, och det var en och annan topp i början sedan höll sig rewarden mellan 0 och 100.



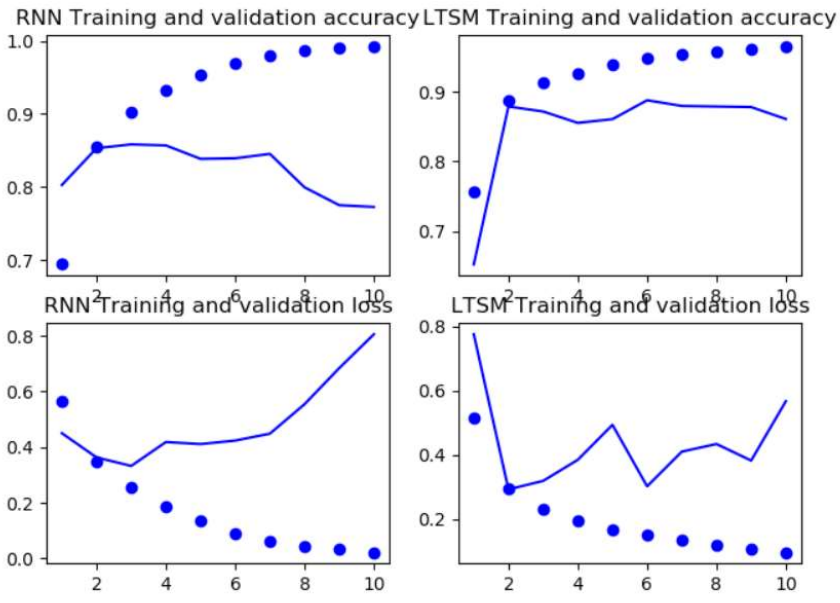
Nedan är en bild av den kumilitativa rewarden, med EPISLON=0,2, GAMMA=0,8, från 0 till 10000 episoder, och man ser att det ger högre reward genom hela.



Natural Language Processing

Source: nlp_basic.py

För NLP testades två olika neurala nätverk med återkopplande egenskaper. Både RNN och LTSM testades, med hjälp av keras SimpleRNN och LSTM lager. För båda neurala nätverken användes data från keras.datasets.imdb vilket består av början av ett antal recensioner och en label antingen positiv eller negativ för varje recension. Båda näten bestod också av ett Embedding lager som gjorde om datan till täta vektorer.
RNN testnogranhet blev 0.78
LTSM testnogranhet blev 0.84
LTSM presterade bättre än RNN nätet.
Training och validation nogrannhet samt training and validation loss visas i bilden nedan.



GAN

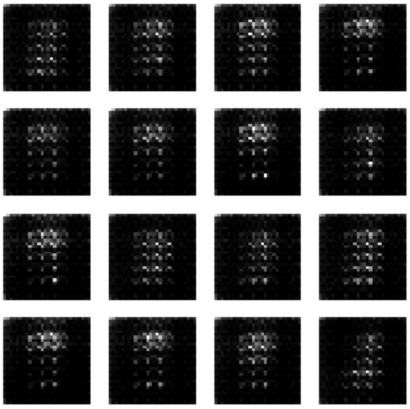
Source: GAN_numbers.py, GAN_numbers_generate_gif.py
Animated gif: dcgan.gif

Figur 1: Brus

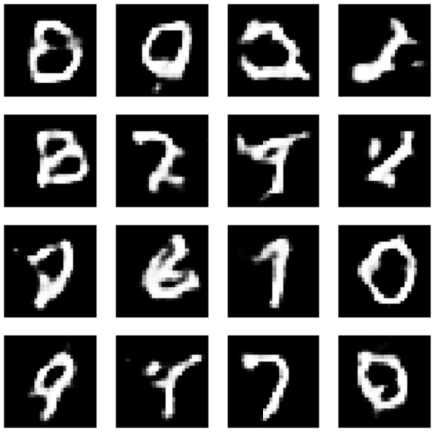
Programmet består av två olika nät, ett som nät konstnären som genererar data från brus, se figur 1 och sen ett annat nät kritikern som bedömer hur bra verk som konstnären genererat. Konstnärens uppgift är att generera siffror som kritikern inte kan skilja bra och dåliga bilder, och kritikern ska bli bättre på att skilja bra och dåliga bilder med hjälp av testdatan. Testdatan består av bilder av handskrivna siffror från minst. Konstnärsnätet består av några olika lager och bland annat av Conv2DTranspose och genererar då en bild från brus. Kritikernätet är ett CNN som klassificera en bild som bra eller dålig, med hjälp av ett positivt eller negativt tal.
En första körning av konstnärsnätet ger en bild enligt figur 2.
Näten tränas sedan om vart annat i 50 iterationer.
Den sista iterationen ger följande resultat för konstnären enligt figur 3, där man ser att bilderna börjar likna siffror även om det i några av fallen är svårt att avgöra vad det är för tal. Och resultatet från kritikern är följande:

```
tf.Tensor([[0.0477845]], shape=(1, 1),
dtype=float32)
```

En animerad gif över alla iterationer som konstnären genererat kan ses i filen dcgan.gif i samma sökväg.



Figur 2: Iteration 1



Figur 3: Iteration 50

Neural Style Transfer

Source: neural_style_transfer.py

Figur 5: Målbild

Figur 4: Stilbild Van Gogh

Neural style transfer går ut på att man har en målbild på ett motiv, sedan har man en stilbild som och man tar då och applicerar stilen från stilbilden på målbilden och får därmed ut en ny bild med samma stil som stilbilden. Man vill då kort använda en lossfunktion som konserverar innehållet men som använder samma stil som stilbilden. Metoden använder olika lager av ett VGG19 nätverk för att beräkna content loss och style loss, där de övre lagren innehåller information som säger mer om bilden och de nedre lagren mer beskriver detaljerna och stilen som används. Sist används en gradient-descent process SciPys L-BFGS-algoritm. Som stilbild används en Van Gogh-målning enligt figur 4. Som målbild används en bild av mitt hus enligt figur 5. Resultatet blev slutbilden i figur 6.

```
Start of iteration 16
Current loss value: 39884110.0
Image saved as style_transfer_at_iteration_16.png
Iteration 16 completed in 201s
Start of iteration 17
Current loss value: 38711730.0
Image saved as style_transfer_at_iteration_17.png
Iteration 17 completed in 216s
Start of iteration 18
Current loss value: 37534796.0
Image saved as style_transfer_at_iteration_18.png
Iteration 18 completed in 204s
Start of iteration 19
Current loss value: 36469360.0
Image saved as style_transfer_at_iteration_19.png
Iteration 19 completed in 204s
```

Figur 6: Slutbild