Artificial Intelligence 2018: Build the Most Powerful AI

SuperDataScience Team

Abstract

Recently, we discovered that a very new kind of AI was invented.The kind of AI which is based on a genius idea and that you can build from scratch and without the need for any framework.We checked that out, we built it, and... the results are absolutely insane! This game-changing AI called Augmented Random Search, ARS for short.

And in a very simple implementation, it is able to do an exact same thing that Google Deep Mind did in their accomplishment last year - which is to train an AI to walk and run across a field.

However, ARS is 100x times faster and 100x times more powerful.

- Be prepared for the most significant tech challenges of the 21st century
- No need for sophisticated algorithms and frameworks
- What Facebook or Google spent on millions or even more - you can literally do at home!
- You will be able to compete with multi-billion dollars companies
- Change the world on your own within months or even weeks
- Build the most powerful AI that anyone has ever built

Also we would like to thank all of the students who have contributed to the discussions and Q&A sections. It's always a great idea and method of enhancing the materials to be able to discuss key concepts. In the following PDF you will see some of the main questions that we see frequently asked. If you don't see the answer that you are looking for please post in the Q&A in the course. In addition, the following section has some of the main setup and general questions for the entire course:

- **Where can I access the course materials and recommended readings?** You can visit the following link to obtain the materials

  `https://www.superdatascience.com/ars/`

  .
- **Which Python version does the course use?** The course was originally built with Python 3.6.

Table of Contents

Summary

1 Section: 1 - Introduction

1.1 Lecture 1. Introduction

- **I received a modulenotfound error for Numpy, how can I resolve it?** You can use the following command to install Numpy: conda install -c anaconda numpy

1.2 Lecture 2. Where to Get the Materials

- **If you have any issue downloading the required material please reach out to the course TA or post in the QA section for assistance.**

2 Section: 2 - Part 1 - Augmented Random Search (ARS) Intuition

2.1 Lecture 3. Plan of attack

- **Is there any general information on types of random search?** Sure, please visit the following for some further context on random search: `https://en.wikipedia.org/wiki/Random_search` but we are focused on groundbreaking technology in this course - Augmented Random Search: `https://arxiv.org/abs/1803.07055`

3 Lecture 4. Overview of Augmented Random Search (ARS)

- **The agent takes more than one discreet action at each time step t, so a group of actions is returned instead of only one. If you work on a model where there would only be one discreet action instead of many, would this type of AI work? Or is it better to look at other types?** Yes, of course, and that's what we do in our other course, "Artificial Intelligence A-Z", where our inputs are both encoded vectors and images, and our output is a single discreet action.

3.1 Lecture 5. Updates on Udemy Reviews

- **Will this course have different face recognition techniques with deep learning?** In this course we cover face detection in the first module, in the second using SSD for detection and GAN's in the third module. You can visit `https://www.superdatascience.com/computer-vision/` for the recommended readings and course materials as well but you can also use the first module and extend it or the SSD for further detection related algorithms.

3.2 Lecture 6. How does a perceptron work?

- **Is the perceptron a neural network with zero hidden layers?**Normally the perceptron is single layer or multi layer, for example a single-layer perceptron network consists of one or more artificial neurons in parallel. With our perceptron example we have a simple single layer perceptron (we also increase the output and make it more complex in the video as well).

3.3 Lecture 7. Maximising Rewards

3.4 Lecture 8. Method of Finite Differences

- **What is meant by Delta Perturbations?** It's basically referring to small values in our matrix of weights with the perturbations (a set of random/small values added and subtracted from the weights) If you get a chance the following might help clarify the use further: `https://arxiv.org/pdf/1803.07055.pdf`and `http://web.mit.edu/16.90/BackUp/www/pdfs/Chapter14.pdf`

3.5 Lecture 9. Basic vs Augmented Random Search

- **Does the 1st augmentation of BRS work because the magnitude of scaling is proportional to how erratic the rewards are? If that's true, the the reason behind it is that we would take a large step if our rewards are super erratic because that indicates that we still have a long way to convergence?) that can't be predicted.**

  To understand the scaling and std of rewards I think the clearest example is to examine section 3.1 especially:

  "To address the large variations of the differences r(M+) r(M), we scale the update steps by the standard deviation R of the 2N rewards collected at each iteration (see Line 7 of Algorithm 2). To understand the effect of scaling by R , we plot standard deviations R obtained during training a policy for the Humanoid–v1 model in Figure 1. The standard deviations R have an increasing trend as training progresses" but yes in that sense the first augmentation could work.

  For part 2, thats why we would scale by the std for example:

  "The standard deviations R have an increasing trend as training progresses This behavior occurs because perturbations of the policy weights at high rewards can cause Humanoid–v1 to fall early, yielding large variations in the rewards collected. Therefore, without scaling by R , our method at iteration 300 would be taking steps which are a thousand times larger than in the beginning of training. The same effect of scaling by R could probably be obtained by tuning a ste the amount of tuning required, and thus we opted for the scaling by the standard deviation."

3.6 Lecture 10 - ARS vs other AI

- **Why the name augmented?** It is called "augmented random search" to highlight the evolution since "basic random search", which was an already existing technique of policy optimization (see the Intuition Lecture "Augmented vs Basic").

4 Section 3 - Part 2 - Augmented Random Search (ARS) Practical

4.1 Lecture 11 - ARS - Step 1

- **Does ARS have better results then other AI algorithms like a3c or is it only faster than others?** It is fast but it depends on the type of algorithm that you are trying to build. For example the main paper introducing ARS states "Computationally, our random search algorithm is at least 15 times more efficient than the fastest competing model-free methods on these benchmarks. We take advantage of this computational efficiency to evaluate the performance of our method over hundreds of random seeds and many different hyperparameter configurations for each benchmark task. Our simulations highlight a high variability in performance in these benchmark tasks, suggesting that commonly used estimations of sample efficiency do not adequately evaluate the performance of RL algorithms."

4.2 Lecture 12 - ARS - Step 2

- **What is the reward that is conveyed to the model that "if you achieve this action you will get this reward"?** The reward is defined by the PyBullet environment. Basically the more the agent manages to stand up or walk, the more positive is that reward, and the more it falls, the more negative it is.

- **Can AI be trained to find the minimization of a discrete function? For example with a vehicle routing problem or something similar?** Yes it's feasible to train with reinforcement learning to solve vehicle routing, for instance: https://arxiv.org/abs/1802.04240.

4.3 Lecture 13 - ARS - Step 3

- **Howcome we're sampling by a Gaussian distribution and why do we need the std in the future?** We are using noise for the sigma in the Gaussian distribution and we are using it to sample the perturbations following the Gaussian/std signal, but for some additional reference i'd like to relate it to the paper for ARS for example: https://arxiv.org/pdf/1803.07055.pdf

A primitive form of random search simply computes a finite difference approximation along the random direction and then takes a step along this direction without using a line search. Our method

ARS, described in Section 3, is based precisely on this simple strategy. For updating the parameters θ of a policy π_θ, our method exploits update directions of the form:

$$\frac{r(\pi_{\theta+\nu\delta}, \xi_1) - r(\pi_{\theta-\nu\delta}, \xi_2)}{\nu}, \tag{2}$$

for two i.i.d. random variables ξ_1 and ξ_2, ν a positive real number, and δ a zero mean Gaussian vector. It is known that such an update increment is an unbiased estimator of the gradient with respect to θ of $\mathbb{E}_\delta \mathbb{E}_\xi [r(\pi_{\theta+\nu\delta}, \xi)]$, a smoothed version of the objective (1) which is close to the original objective (1) when ν is small [24]. When the function evaluations are noisy, minibatches can be used to reduce the variance in this gradient estimate. The basic random search (BRS) algorithm is outlined in Algorithm 1. Evolution Strategies is version of this algorithm with several complicated algorithmic enhancements [28]. BRS is called Bandit Gradient Descent by Flaxman et al. [7]. We note the many names for this algorithm, as it is at least 50 years old and has been rediscovered by a variety of different optimization communities.

4.4 Lecture 14 - Checkpoint!

```
Congrats on reaching this first checkpoint!
Here is the code we have implemented so far:

# AI 2018

# Importing the libraries
import os
import numpy as np

# Setting the Hyper Parameters

class Hp():

    def __init__(self):
        self.nb_steps = 1000
        self.episode_length = 1000
        self.learning_rate = 0.02
        self.nb_directions = 16
        self.nb_best_directions = 16
        assert self.nb_best_directions <= self.nb_directions
        self.noise = 0.03
        self.seed = 1
        self.env_name = ''
Until then, enjoy AI!
```

4.5 Lecture 15 - ARS - Step 4

- **Normalizing means to "Standardize" the inputs, to make the inputs have a mean of zero and a standard deviation of 1, correct?** That's correct, the only tricky part is that we are doing it at each step when reaching a new step so it has to be what we call an online normalization of the states.

4.6 Lecture 16 - ARS - Step 5

- **Why we need to compute mean dif.?** We need to use it from the main setup from the ARS research paper - in order to improve the performance of the ARS algorithm we have to normalize the states, in order to normalize the values we will need to subtract by the mean and divide by the standard deviation, basically it will help the neural network having values between 0 - 1.

- **Would x just be a vector describing our cheetah's placement in the environment like [23, 1, 55, ...]?**

  Correct, our x is the value of our state so for example when using:

  ```
  self.mean_diff += (x - last_mean) * (x - self.mean)
  ```

  We are computing it with the self.mean and updating our computation for the mean applied to our state.

4.7 Lecture 17 - ARS - Step 6

- **Could you elaborate about the function of returning to this code and what this code means by subtracting the mean and divided by the standard deviation: return (inputs - obs_mean) / obs_std**

  That part of the code is basically to return our input to normalize our states that we have ready from:

  ```
  obs_mean = self.mean

  obs_std = np.sqrt(self.var)
  ```

  Also for normalizing the following might add some further context: https://www.udemy.com/artificial-intelligence-ars/learn/v4/questions/4604794

4.8 Lecture 18 - Checkpoint!

Congrats on reaching this new checkpoint!
Here is the code we have implemented so far:

```
# AI 2018

# Importing the libraries
import os
import numpy as np

# Setting the Hyper Parameters

class Hp():

    def __init__(self):
        self.nb_steps = 1000
        self.episode_length = 1000
        self.learning_rate = 0.02
        self.nb_directions = 16
        self.nb_best_directions = 16
        assert self.nb_best_directions <= self.nb_directions
        self.noise = 0.03
        self.seed = 1
        self.env_name = ''

# Normalizing the states

class Normalizer():

    def __init__(self, nb_inputs):
        self.n = np.zeros(nb_inputs)
        self.mean = np.zeros(nb_inputs)
        self.mean_diff = np.zeros(nb_inputs)
        self.var = np.zeros(nb_inputs)

    def observe(self, x):
        self.n += 1.
        last_mean = self.mean.copy()
        self.mean += (x - self.mean) / self.n
        self.mean_diff += (x - last_mean) * (x - self.mean)
        self.var = (self.mean_diff / self.n).clip(min = 1e-2)

    def normalize(self, inputs):
        obs_mean = self.mean
        obs_std = np.sqrt(self.var)
        return (inputs - obs_mean) / obs_std
```

Until then, enjoy AI!

4.9 Lecture 19 - ARS - Step 7

4.10 Lecture 20 - ARS - Step 8

- **What is "Input"** An input is two folds: 1. It is something that describes exactly what is happening in the environment at each time t. 2. It is what will be fed to the neural network (a perceptron in the course) before the latter returns the output.
- **Are inputs and states the same?** Inputs and states are the same. That's why in the course you will see them referred to as " the input states".

5 Section 21 - ARS - Step 9

5.1 Lecture 22 - ARS - Step 10

5.2 Lecture 23 - Checkpoint!

5.3 Lecture 24 - ARS - Step 11

5.4 Lecture 25 - Checkpoint!

5.5 Lecture 26 - ARS - Step 12

5.6 Lecture 27 - ARS - Step 13

5.7 Lecture 28 - ARS - Step 14

5.8 Lecture 29 - ARS - Step 15

5.9 Lecture 30 - ARS - Step 16

5.10 Lecture 31 - ARS - ARS - Step 17

5.11 Lecture 32 - ARS - Step 18

5.12 Lecture 33 - ARS - Step 19

5.13 Lecture 34 - Checkpoint!

5.14 Lecture 35 - ARS - Step 20

- **I viewed the source code on github for http://gym.openai.com/envs/Humanoid-v2/ but I didn't see anything in that code that gives me what I need. That source code appears to pull from mujoco_env, do**

you know of any mujoco environment source code I could reference? I think you can see the environment code more in depth in the "algorithmic_env" file, which you can find in the following path:

5.15 Lecture 36 - For Windows users only

5.16 Lecture 37 - ARS - Final Results

- **You mentioned at the beginning of the course that ARS could be used to tackle business problems. Can you give an example of how you'd do that?** There was a recent discussion relating to ARS in other environments/situations that I think will help: `https://www.udemy.com/artificial-intelligence-ars/learn/v4/questions/4628440`
- **Are there any other environments to experiment on?** Yes, please see the following: `https://www.udemy.com/artificial-intelligence-ars/learn/v4/questions/4615792`.

6 Section 6 - Module 2 - Object Detection with SSD

6.1 Lecture 27 - Object Detection - Step 1

- **I'm receiving the following error: ValueError: not enough values to unpack (expected 2, got 0). What can I do to resolve it?** Please open the terminal or prompt and if you are using a virtualplease activate it with:

```
Mac/Linux:
Source Activte my\_env\_name

Windows
Activate my\_env\_name

Then please run pip list or conda list to check the version of PyTorch.
If it's 0.4.0 (which has introduced this bug)
to resolve it you can downgrade to any version < 0.4.0.

For Mac/Linux you can find the versions here:
https://pytorch.org/get-started/previous-versions/

For Windows:

# If your main Python version is not 3.5 or 3.6
conda create -n test python=3.6 numpy pyyaml mkl

# for CPU only packages
conda install -c peterjc123 pytorch-cpu

# for Windows 10 and Windows Server 2016, CUDA 8
conda install -c peterjc123 pytorch

# for Windows 10 and Windows Server 2016, CUDA 9
conda install -c peterjc123 pytorch cuda90

# for Windows 7/8/8.1 and Windows Server 2008/2012, CUDA 8
conda install -c peterjc123 pytorch_legacy
```

- **I'm receiving the following error: ImportError: DLL load failed: The specified procedure could not be found. What can I do to resolve it?** For the DLL error, it's normally caused by the version of PyTorch. If you don't have Nvidia drivers please re-install PyTorch with

the CPU version. If it persists, please follow the steps listed here: `https://pytorch.org/docs/stable/notes/windows.html#import-error`.

6.2 Lecture 28 - Object Detection - Step 2

- **Why are we using PyTorch over Tensorflow?** Using PyTorch helps considerably speed up the training computations, and the predictions as well. Comparing it to Keras, it is way faster, more efficient and more powerful. It can also come down to architecture choice and PyTorch can be easier to work with at times to implement specific modules such as the SSD.

- **I'm receiving an error stating the OpenCV module can't be found. What steps can I use to resolve it?**

  If you are using the virtual platform please activate it first with:

  Mac/Linux

  ```
  source activate virtual_platform
  ```

  Windows

  ```
  activate virtual_platform
  ```

  Then please run:

  ```
  pip install opencv-python
  ```

6.3 Lecture 29 - Object Detection - Step 3

- **What does transforming the image to be compatible with the Neural Network mean?**It's referring to convert the size of the images, put them in the right dimensions, the right scales of colour values, according to the convention under which the neural network was trained.

6.4 Lecture 30 - Object Detection - Step 4

- **I received an error stating no module named 'data'. How can I resolve it?** Please double check the working environment includes all of the files such as discussed here: `https://www.udemy.com/computer-vision-a-z/learn/v4/questions/4172716`.

- **What do we use the unsqueeze function for?**We use the unsqueeze function to create the fake dimension of the batch (our third transformation) since the neural network can't accept a single input image because it only accepts batches of inputs. When you see unsqueeze in a neural

network it's before fitting the nn with input. For our argument we need to use the index of the dimension of the batch (0). You can see some further information at: http://pytorch.org/docs/master/tensors.html?highlight=unsqueeze#torch.Tensor.unsqueeze.

- **Why are we changing numpy array into tensor variable?**Converting it to a torch tensor allows us to work with a more useful (due to it's matrix setup) with a neural network compared to a numpy array and allows us to use the methods we further implement. It can always come down to development technique as well, but when working with a neural net, pytorch tensors due have a big benefit, for example for some additional context: http://pytorch.org/docs/master/tensors.html.

- **Is there any other information on realtime detection?**Please take a look at the following discussion for some info on real time detection: https://www.udemy.com/computer-vision-a-z/learn/v4/questions/3513956 and also at https://www.udemy.com/computer-vision-a-z/learn/v4/questions/342544.

6.5 Lecture 31 - Object Detection - Step 5

- **What is the intutition behind normalization?**It's due to the need to specify the dimensions of the tensor (and the locations of the rectangle) to normalize the scale of objects/values of the detected objects. Our new tensor object that has 4 dimensions w/h/w/h is created due to the position of the detected objects within the image has to be normalized between 0 - 1. This tensor (scale) normalizes the objects detected in the image. W/H is for the upper left hand corner of the rectangle detector and the second W/H for the lower right for the same rectangle detector.

  scale = Torch.tensor([Width, Height, Width, Height])

  As for using the boxes, have a look at this additional reference when you get the chance :

  https://www.learnopencv.com/how-to-select-a-bounding-box-roi-in-opencv-cpp-python/.

6.6 Lecture 32 - Object Detection - Step 6

- **At code line 20, which is a comment, could you share more clarification as to understand what is "the batch"?**The tensor contains 4 elements (the first being the batch) and it was created with unsqueeze and is our outputs. For some additional information and examples see the following:

https://www.quora.com/What-is-meant-by-Batch-in-machine-learning
and
https://stackoverflow.com/questions/41175401/what-is-a-batch-in-tensorflow.

6.7 Lecture 33 - Object Detection - Step 7

- **I didn't fully understand the line: map_location = lambda storage, loc: storage. Can you please elaborate further on it?**. The map_location = lambda storage, loc: storage is used to remap the storage of all tensors into the CPU. If you would like some additional information please see the following: https://discuss.pytorch.org/t/on-a-cpu-device-how-to-load-checkpoint-saved-on-gpu-device/349.

6.8 Lecture 34 - Object Detection - Step 8

-

6.9 Lecture 35 - Object Detection - Step 9

- **Please see the following for information on using CUDA for the SSD:** https://www.udemy.com/computer-vision-a-z/learn/v4/t/lecture/8127652?start=0.

6.10 Lecture 36 - Object Detection - Step 10

- **I received the error: RuntimeError: sum() missing 1 required positional arguments: "dim". How can I resolve it?**Please try changing the following:

  Change
  ```
  num_pos = pos.sum(keepdim=True)
  ```
  To
  ```
  num_pos = pos.sum(dim=1, keepdim=True)
  ```

6.11 Lecture 37 - Training the SSD

6.12 Quiz 4 - Object Detection with SSD

7 Section 7 - Homework Challenge - Detect Epic Horses galloping in Monument Valley

7.1 Lecture 38 - Homework Challenge - Instructions

7.2 Lecture 39 - Homework Challenge - Solution (Video)

7.3 Lecture 40 - Homework Challenge - Solution (Code files)

7.4 Section 8 - Module 3 - Generative Adversarial Networks (GANs) Intuition

7.5 Lecture 41 - Plan of Attack

7.6 Lecture 42 - The Idea Behind GANs

- **Is there a main difference between SSDs and GANs?**

  SSD –

  Key idea here is single network (for speed) and no need for region proposals instead it uses different bounding boxes and then adjust the bounding box as part of prediction. Different bounding box predictions is achieved by each of the last few layers of the network responsible for predictions for progressively smaller bounding box and final prediction is union of all these predictions.

  SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage.

  https://medium.com/@ManishChablani/ssd-single-shot-multibox-detector-explain

  GANs –

  Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other (thus the adversarial).

  GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014. Referring to GANs, Facebooks AI research director Yann LeCun called adversarial training the most interesting idea in the last 10 years in ML.

  GANs potential is huge, because they can learn to mimic any distribution of data.

```
That is, GANs can be taught to create worlds
eerily similar to our own in any domain: images,
music, speech, prose. They are robot artists in
a sense, and their output is impressive, poignant even.

https://deeplearning4j.org/generative-adversarial-network
```

```
If you want to look at some similar algorithms to SSD,
you could compare it to the use of YOLO, YOLO2/3 or an
RNN, but each has their own advantages and disadvantages
related to the use case.
```

7.7 Lecture 43 - How Do GANs Work? (Step 1)

- **Why does the discriminator need to learn on the new generated images?** This reference does a great job of explaining the information to help clarify that question and it has a Spongebob reference which is always a plus:

  `https://medium.com/@awjuliani/generative-adversarial-networks-explained-with-a-classic-spongebob-squarepants-episode`

  That being said a key take away from that article is that: "Through training, the discriminator network learns a function to tell the difference between the real and generated data. The first features the discriminator learns to look for may be relatively obvious aspects of the data which easily separate the real from the fake"

7.8 Lecture 44 - How Do GANs Work? (Step 2)

7.9 Lecture 45 - How Do GANs Work? (Step 3)

- **Are the same real dog images shown to the Discriminator on each 'step/round' for the Discriminator to compare with the Generated ones or always different from a huge library of real dog images ?** ""A Generative Adversarial Network works through the interplay between two semi-separate networks: a generator and a discriminator. The goal of the discriminator is to tell the difference between the data generated by the generator and the real-world data we are trying to model. We can think of the discriminator as being like the bouncer at a club. I don't have just any bouncer in mind though. Discriminator networks are like the

bouncer outside the Salty Spitoon, as seen in the Spongebob Squarepants episode:http://spongebob.wikia.com/wiki/No_Weenies_Allowed.

In the world of Spongebob Squarepants, one must be tough in order to get into the Salty Spitoon. The job of the bouncer (discriminator) is to tell the difference between the real tough guys (the real data), and the weenies (the generated data).

Then we have Spongebob Squarepants (the generator). He certainly isn't a real tough guy. He is an imitator, and needs to learn to look like a tough guy in order to get into to see his friend."...

If you get the chance, take a look for additional information here:

https://medium.com/@awjuliani/
generative-adversarial-networks-explained-with-a-classic-spongebob-squarepants-episode

7.10 Lecture 46 - Applications of GANs

7.11 Quiz 5: Generative Adversarial Networks (GANs) Intuition

8 Section 9 - Module 3 - Image Creation with GANs

8.1 Lecture 47 - GANs - Step 1

- **Why did we define the weight initialization function? Also, what is the default value of initialization function in PyTorch?** We use the weight initialization so that it will take a nn to initialize the weights of the nn (applied to both the nn of the generator and the nn of the discriminator) for the adversarial networks.As for the default initialization the following will provide some further context for PyTorch:

  https://discuss.pytorch.org/t/whats-the-default-initialization-methods-for-layers/3157 and also

  https://discuss.pytorch.org/t/how-are-layer-weights-and-biases-initialized-by-default/13073.

8.2 Lecture 48 - GANs - Step 1

- **What is the meaning of .bias.data.fill_(0)?** This is used to initialize the bias within the linear layer of the neural network. For example please see these two additional examples as they will help provide some further context:

  https://discuss.pytorch.org/t/how-are-biases-initialized-in-pytorch-in-the-linear-layer/5049/5 and also https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/linear.py#L44-L48.

- **Could you please share papers based on which this architecture is adopted and implemented?** The architecture created in Module 3 is inspired from this paper: https://arxiv.org/pdf/1511.06434.pdf.

8.3 Lecture 49 - GANs - Step 2

- **How we can pass input to a Class G? Is that really possible to pass arguments like that?**Yes it is because the forward function is connected to the __init__ function via "self". In that case you don't have to specify the name of the method. But in general case you do have to specify the name of the method following the object and a dot, like this:

```
output = netG.forward(input)
```

8.4 Lecture 50 - GANs - Step 3

8.5 Lecture 51 - GANs - Step 4

- **Is the article of experiment of the figures available?** It's from the following paper: `https://arxiv.org/pdf/1511.06434.pdf`.

8.6 Lecture 52 - GANs - Step 5

8.7 Lecture 53 - GANs - Step 6

8.8 Lecture 54 - GANs - Step 7

8.9 Lecture 55 - GANs - Step 8

8.10 Lecture 56 - GANs - Step 9

- **Why don't we use netD.foward to get the output?** It is being forward propagated, we are feeding the nn with the input (a mini batch of real images) and through the main module that is forward propagating the real images inside the mini batch -

```
output = netD(input) # We forward propagate this real
image into the neural network of the discriminator to get the
prediction (a value between 0 and 1).

output = netD(fake.detach()) # We forward propagate the fake generated
images into the neural network of the discriminator to get
the prediction (a value between 0 and 1).
```

- **What do you mean a tensor also has attached gradients?** We need to (in the discriminator) take the nn to use zero_grad() that automatically initializes the gradients with respect to the weights.

  The following might help with tensors and gradients: `https://pytorch.org/tutorials/beginner/examples_autograd/two_layer_net_autograd.html`.

8.11 Lecture 57. GANs - Step 10

8.12 Lecture 58. GANs - Step 11

- **Can I run the GAN with CUDA?** Yes, you can run it with CUDA. Please see the following for some further information: `https://www.udemy.com/computer-vision-a-z/learn/v4/questions/3096984`

8.13 Quiz 6: Image Creation with GANs

8.14 Lecture 60 - 60. Special Thanks to Alexis Jacq

9 Section 10 - Annex 1: Artificial Neural Networks

9.1 Lecture 61 - What is Deep Learning?

9.2 Lecture 62 - Plan of Attack

9.3 Lecture 63 - 63. The Neuron

9.4 Lecture 64 - 64. The Activation Function

- **Could you provide any other information on using sigmoid functions for neural nets?** Absolutely, please see the following useful discussion:

  `https://stats.stackexchange.com/questions/162988/why-sigmoid-function-instead-of-anything-else`

9.5 Lecture 65 - How do Neural Networks work?

9.6 Lecture 66 - How do Neural Networks learn?

9.7 Lecture 67 - Gradient Descent

9.8 Lecture 68 - Stochastic Gradient Descent

9.9 Lecture 69 - Backpropagation

- **Can you provide any further information for the learning rate?** Learning rate does play an important role:"Learning rate is a hyperparameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope. While this might be a good idea (using a low learning rate) in terms of making sure that we do not miss any local minima, it could also mean that we'll be taking a long time to converge—especially if we get stuck on a plateau region" `https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in` addition, see the following on learning rate in Neural Nets (Back Propagation) by Andrew Ng:`http://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf`

10 Section 11 - Annex 2: Convolutional Neural Networks

10.1 Lecture 70 - Plan of Attack

10.2 Lecture 71 - What are convolutional neural networks?

10.3 Lecture 72 - Step 1 - Convolution Operation

10.4 Lecture 73 - Step 1(b) - ReLU Layer

10.5 Lecture 74 - Step 2 - Pooling

10.6 Lecture 75 - Step 3 - Flattening

10.7 Lecture 76 - Step 4 - Full Connection

10.8 Lecture 77 - Summary

10.9 Lecture 78 - Softmax & Cross-Entropy

11 Conclusion

Please check back often as we will continuously update the FAQ document, and if you have any questions please feel free to post in the course Q&A.