0.   
스티브 워즈니악 옹은 "All my great stuff, I learned outside of school"이라 했다.

누가 가르쳐 주는 것을 그대로 배우는 것 보다, 스스로 여기저기 헤매 보면서 이것저것 생각 해 보고 만들어 보는 것이 가장 좋다고 생각한다. 나는 그렇게 헤매면서 여기까지 왔고, 앞으로도 열심히 즐겁게 헤매고 다닐것 같다.

하지만 헤매고 있을 시간이 없는 경우, 또는 헤매는 것이 즐겁지 않고 짜증만 나는 경우에는 먼저 간 사람이 찾아둔 길을 따라가는 것이 좋겠다. 내가 찾아둔 지름길을 공유 한다.

1.   
가장 먼저 할 일은 코세라의 앤드류 응 교수님 강의를 듣는 것이다.

강의 동영상은 유튜브에도 모두 공개되어 있다. 하지만 꼭 코세라에서 들으면서 옥타브로 숙제를 해 봐야 한다. 무료로도 들을 수 있지만, 79달러를 내고 들으면 강의 내용이 머리 속에 더 잘 들어 오게 된다. 진짜다.

\* 코세라의 앤드류 응 교수님의 머신러닝 강의: <https://www.coursera.org/certificate/machine-learning>

\* 다른 스타일의 강의들 모음: <https://www.facebook.com/dgtgrade/posts/1145680742157457>

한국어 강의를 원하면 김성훈 교수님의 [모두를 위한 딥러닝] 강의를 들으면 된다.

\* 모두를 위한 딥러닝: [https://youtu.be/BS6O0zOGX4E…](https://youtu.be/BS6O0zOGX4E?list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm)

혹시 그냥 엄청나게 쉬운 강의부터 시작하고 싶으면 휴먼러닝을 들으면 된다.

\* 휴먼러닝: [https://youtu.be/CRaKdnk2W9c…](https://youtu.be/CRaKdnk2W9c?list=PLefQdA1SdkhtRUuN_D3PdxaR2XTGQw8Ph)

2.   
1번의 강의를 듣다 보면 금새 선형회귀와 신경망의 기초에 대해서 알게 된다. 이때 Universal Approximation Theorem에 대해서 공부한 뒤 간단한 신경망을 직접 Python+Numpy로 구현해 봐야 한다.

\* 참고 글: [https://www.facebook.com/groups/TensorFlowKR/permalink/332680743739657](https://www.facebook.com/groups/TensorFlowKR/permalink/332680743739657/)

꼭 Python으로 해야 하는 건가? 아니다. 그럴리가 없지 않은가. 하지만 내가 추천하는 지름길은 Python이다. 그게 대세이기 때문이다. 대세가 꼭 나나 너에게 맞는 것은 아니지만 대세를 따르면 편하다. 이글은 어디까지나 지름길에 대한 글이지 나의 길이나 너의 길에 대한 글이 아니다. Python이 생소하더라도 괜찮다. Python은 초기 진입 장벽이 매우 낮은 언어이므로 겁먹을 필요 없다.

처음에는 Numpy의 신택스가 좀 어려워 보일 수 있는데 익숙해지고 나면 Numpy 없으면 못 살거 같은 느낌이 들 정도로 Numpy는 멋진 것이다.

Python+Numpy+기타 등등의 설치의 지름길은 Anaconda다. IDE는 나는 PyCharm을 사용하고 있다. 원래 vi를 사랑하고 (진짜로) 지금도 사랑하지만 PyCharm 써보고 나니 앞으로 적어도 Python 코딩은 vi에서는 못 할 것 같다.

개발 환경 준비에 대한 더 상세한 내용은 아래의 영상을 참고하면 된다.

\* 머신러닝 개발환경 준비: [https://youtu.be/pMkwjXFZdH4…](https://youtu.be/pMkwjXFZdH4?list=PLefQdA1SdkhtRUuN_D3PdxaR2XTGQw8Ph)

3.   
신경망과 Python, 그리고 Numpy가 어느 정도 익숙해지고 나면 드디어 TensorFlow를 사용해 볼 때다.

마찬가지로 꼭 TensorFlow여야 하는가? 하는 질문이 있을 수 있다. 아니다. 하지만 TF가 대세다.

앞에서 지도학습 문제를 풀어봤으므로 이제는 비지도 학습 문제를 풀어볼 때가 되었다. 비지도 학습이란 것을 이해할 수 있는 가장 쉬운 방법은 Auto Encoder를 직접 만들어 보는 것이다.

그리고 데이터는 머신러닝의 헬로월드인 MNIST를 사용하면 된다. 즉, 목표는 MNIST AE를 만드는 것이다.

아직은 CNN에 대해서 모르므로 괜히 어렵게 CNN으로 하지 말고 FC로만 해도 된다. MNIST는 너무 쉬운 문제라서 FC로 해도 잘 된다.

TensorFlow가 너무 빨리 업그레이드 되고, 스펙도 많이 변하고 있어서, 겨우 몆주전에 올라온 글의 정보가 현재 상황과 맞지 않거나, 예제 코드가 지금은 잘 안 돌아가기도 하는데 그 점 주의해야 한다. 웹에서 구한 예제 코드가 바로 안 돌아간다고 겁먹을 필요 없다. 대부분 아주 살짝씩만 수정해 주면 돌아갈거다. 어떻게 수정할 지는 스택오버플로우가 알려줄거다.

그리고 윈도우에서도 TensorFlow 잘 돌아간다. GPU도 잘 이용할 수 있다.

더 심각하게 머신러닝을 열심히 잘 해 보려면 결국 리눅스로 가는게 이래저래 편하긴 하다. TF외의 라이브러리들이 윈도우에서 잘 안 되는 경우들이 간혹 있다. 하지만 일단 이 글에서 다루는 지름길의 범위 내에서는 그냥 윈도우에서 해도 문제 없다.

4.  
여기까지 왔으면 딥러닝이 어떤 것인지 대강 감은 잡힌다. 이제부터 신나게 이것저것 공부하고 만들어 보고 하면 된다. 그런데 역시 시간을 아끼고 싶으면 추가로 아래글도 읽어 보고 길을 떠난다.

\* 참고 글: <https://www.facebook.com/dgtgrade/posts/1328790023846527>

5.  
이제부터는 훨씬 더 큰 규모의 신경망을 다뤄야 하고, 그러려면 ReLU와 친해져야 한다.

ReLU에 대해서 진지하게 고민해 볼 필요가 있다. ReLU가 왜 비선형인지? 뉴런에서 비선형 처리 부분이 없어지면 어떻게 되는 것인지? 왜 Sigmoid 보다 ReLU가 좋은 것인지? 등에 대해서 고민해 본다.

\* 참고 글: <https://www.facebook.com/dgtgrade/posts/1337481972977332>

추가로 Bias가 없으면 어떻게 되는지 등 신경망 전체의 수학적 의미에 대해서 다각도로 고민해 본다.

Sigmoid, Tanh를 사용한 경우 괜히 어렵게 느껴지지만 ReLU를 사용한 경우 신경망 전체 계산에서 곱하기, 더하기, Max 밖에 없어서 어렵게 생각할 것 없다.

머리속에서만 고민해 보지 말고 코딩을 해 보면 좋다. 여기까지 왔으면 이런저런 실험 코딩은 어렵지 않을 거다.

6.   
지금까지 해 본 것보다 훨신 큰 규모의 신경망을 학습 시키기 위해서는 GPU가 필요하다. CPU로 할때 보다 10배~50배 정도 빠르게 학습 시킬 수 있다. CPU로 하면 뭐 하나 돌려보고 드라마 하나 보고 자리에 돌아와 보면 에러 떠 있다. 같은 것을 GPU로 돌려 보면 차한잔 할 새도 없이 에러가 떠 있을 것다. CPU로 하면 돌려 놓고 다음 날 아침이 되어서야 결과를 볼 수 있는 경우가 꽤 있다. 죽는다. 답답해서.

GTX1080이 필요하다. (상세설명 생략하고) 대세다. 문제는 가격인데 대강 100만원 정도 한다. 그리고 1개 보다 2개가 좋다. 학습을 2배로 빨리 할 수 있기 때문만은 아니다. 1개 밖에 없는 경우에는 하나 학습 돌려 놓은 뒤로 추가로 개발 진행한 내용을 잠깐만 실험 삼아 돌려 보는 것을 할 수 없기 때문이다. 아주 할 수 없지는 않을 것 같은데 아무튼 쉽지 않다.

그런데 이 녀석을 2개나 꼽으려면 메인보드와 파워도 좋아야 한다. 그리고 GTX1080은 딥러닝 용으로만 사용하고 디스플레이 용으로는 메인보드의 내장 그래픽 카드를 사용하는 것이 여러모로 편하므로 내장 그래픽 카드도 쓸만해야 한다. 예를 들어 4K 모니터를 사용 중이라면 내장 그래픽카드가 4K를 지원해야 하겠다.

이렇게 해서 준비하면 본체에만 대략 총 300만원 정도의 투자가 필요하다. 학교나 회사 등의 조직에 속한 경우에는 조직장을 잘(...) 설득하여 구매하면 된다. 개인의 경우에는... 그렇다. 이런 투자를 쉽게 하기는 어려울 거다. 문제다.

조직이든 개인이든 당장 이 정도의 투자가 어려운 경우에는 AWS나 Google Cloud 등을 이용하는 것도 방법이겠다. 아무래도 개발은 로컬에서 하는 것이 편할테니 개발은 로컬에서 하고 CPU로만 돌려 본 후에 본격적인 학습은 클라우드에서 GPU로 돌리면 되겠다. 코드 변경은 필요 없고, 클라우드 사용료는 시간당 몇백원 수준이라고 한다.

나는 클라우드에서는 열심히 해 본 적이 없어서 더 상세한 안내는 못 한다.

7.   
이제 드디어 개와 고양이를 분류해 볼 수 있는 때가 왔다. Convolutional Neural Network를 사용하면 된다.

CNN을 공부하고, CNN을 이용해서 무언가 재밌는 걸 만들어 본다. 이렇게 딥러닝이 핫하게 된 것도 CNN 때문이다. CNN으로는 많은 일을 해 볼 수 있다. 수백줄 정도의 TF 코드로만으로도 본인 스스로 놀랄만한 신기한 것들을 만들어 볼 수 있다. 매우 잘 만들면 세상 사람들도 놀래킬 수 있다.

\* 참고 글: [https://www.facebook.com/groups/TensorFlowKR/permalink/341049512902780](https://www.facebook.com/groups/TensorFlowKR/permalink/341049512902780/)

용기 있는 사람들은 이쯤에서 GAN을 해 봐도 된다. GAN은 정말... 기발하고 기특한 녀석이다.

\* 참고 글: [https://www.facebook.com/groups/TensorFlowKR/permalink/420422164965514](https://www.facebook.com/groups/TensorFlowKR/permalink/420422164965514/)

8.  
그런데 막상 이렇게 큰 신경망을 다루려다 보면 이런 저런 이유로 잘 안 될거다. 그러면 이제 Batch Normalization을 적용해야 한다.

BN은 감동이다. 앞에서 얘기한 이런저런 문제들 중 기초적인 문제들을 한방에 해결해 준다.

\* 참고 글: <https://www.facebook.com/dgtgrade/posts/1337790532946476>

9.   
이제 강화학습을 공부해 볼 수 있다. 알파고는 바로 이 강화학습으로 만들어졌다. 강화학습은 OpenAI Gym에서 진행하는 것이 좋다.

이쯤 되면 더 이상 이런 단편적인 내용의 글이 필요할 것으로 생각하지 않는다.

10.   
각자의 길을 간다.

다만 남들이 고민해 둔 거 그대로 공부만 하려 하지 말고, 또 공부 많이 한 다음 뭔가 만들어 보려 하지 말고, 간단한 뭔가라도 실제로 만들어 보면서 공부 하기를 추천한다.

물론 대부분의 경우 찾아보면 분명히 남들이 이미 더 잘 정리 하고 잘 만들어 둔 것이 있을 거다. 하지만 본인이 직접 만들어 보면서 1) 문제를 발견 하고 2) 해결책을 고민해 보고 하는 것이 남들이 정리해 둔 문제와 해답을 먼저 보는 것 보다 훨씬 공부가 많이 된다.

특히 답이 아니라 문제가 뭔지를 스스로 발견하는 과정은 매우 중요하다.