**CIFAR10 인식 정확도 챌린지**

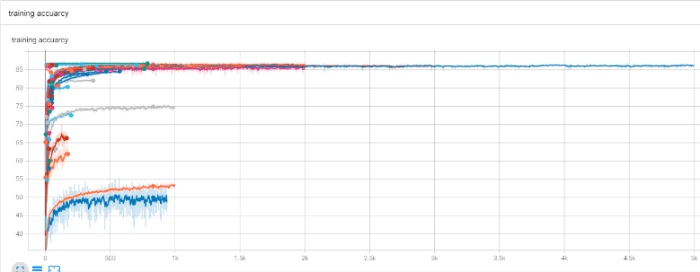
김준호 201710860

최종 정확도 : 86.8056%

* **전략**

1. 교수님께서 먼저 만들어 주신 모델을 base model로 삼고 여기서 바꿀 수 있는 것들을 하나하나 바꿔보며 훈련 진행 과정과 결과를 살펴본다.
2. colab, AWS는 훈련속도가 느리므로 개인 GPU를 이용하여 학습한다.
3. tensorboard를 이용하여 매 epoch마다 loss와 validation set의 accuracy를 저장해놓고 다른 모델과 비교한다.
4. 더 좋은 모델이 있다면 해당 모델을 다시 수정한다.
5. 최고 accuracy를 갱신하면 해당 weights를 저장한다.

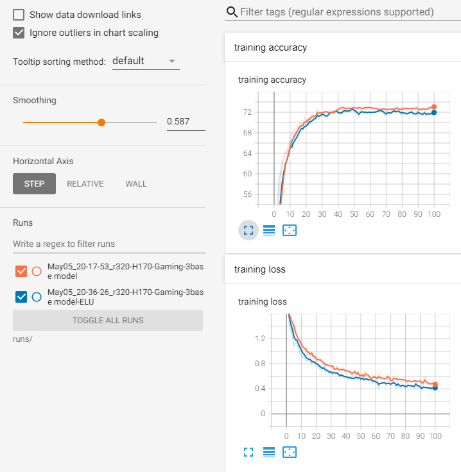
* **시도의 흔적들**

****

이 중에서 정확도를 크게 높였던 부분을 정리하겠다.

* **Activation function**

ReLU 함수를 ELU로 바꾼다.

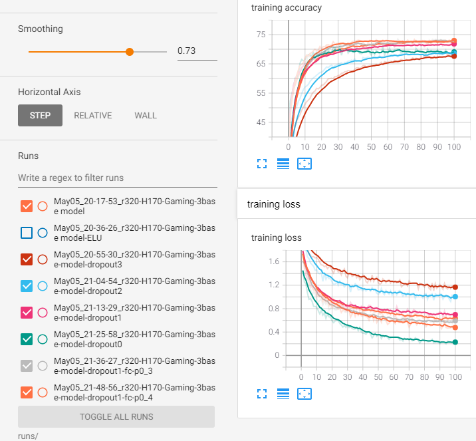


거의 차이가 없지만 ReLU가 조금 더 나은 정확도를 보인다.

* **Drop out**

dropout의 확률 p는 0.5로 통일한다.

fully connected layer에서는 droup out을 모두 사용하고 convolution layer에서는 낮은 단계에서부터 하나씩 빼며 실험을 진행한다.

****

dropout층을 convolution layer에서는 마지막에 하나, fc layer에서는 확률을 0.4, 0.3으로 주는 것이 조금 나았다.

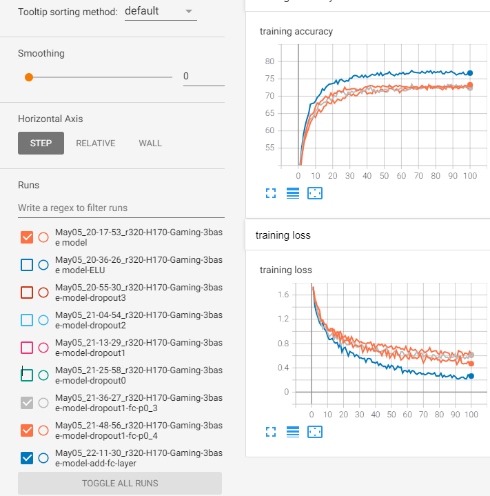
* **fully connected layer의 capacity 증가**

정확도의 한계가 모델이 작아서 그런 것이 아닐까 생각해서 이에 fc 층을 아래와 같이 변경하였다.

[64 \* 8 \* 8, 100] -> [100, 10] 을

[64 \* 8 \* 8, 64 \* 4 \* 4] -> [64 \* 4 \* 4, 64 \* 2 \* 2] -> [64 \* 2 \* 2, 10]

위와 같이 바꾸어서 dropout의 최종 결과인 convolution layer에는 dropout을 마지막에만, fc layer에는 p=0.4로 모두 넣었다.

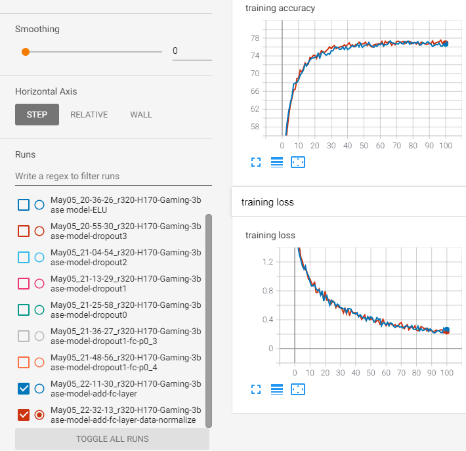


정확도가 약 5% 증가한 것을 볼 수 있다.

* **CIFAR10 데이터의 data normalization**

<https://github.com/facebookarchive/fb.resnet.torch/issues/180>

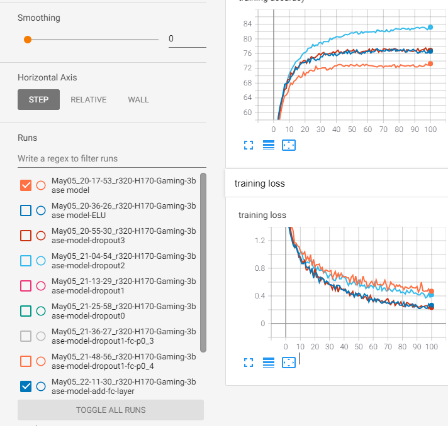
위 github repository에서 값을 참고하였다.



거의 차이가 보이지 않는다.

* **Data Augmentation**

data augmentation도 여러가지 시도를 해보았지만 horizontalflip과 randomrotation을 조금 주는 것이 가장 좋았었다.



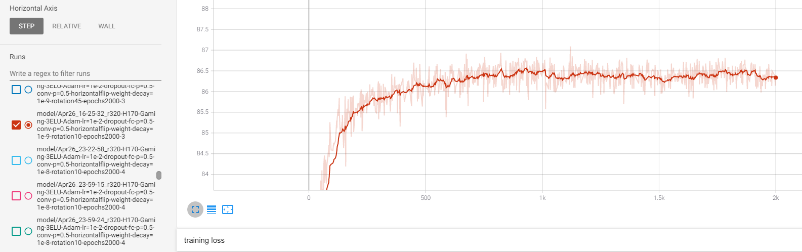
위 처럼 100epoch이었음에도 불구하고 우상향하는 그래프를 볼 수 있으므로 1000 epochs train 하겠다.

* **Final (ELU, lr=1e-2, weight\_decay=1e-9, batchnorm layer와 dropout layer swap)**

최종적으로는 activation function으로는 ELU, dropout의 확률은 모두 0.5, weight-decay를 1e-9를 준 방법으로 했을 때 성능이 가장 좋았다. 또한 learning rate를 1e-2로 하였다.

또한 convolution layer의 BatchNormalization층과 dropout층의 위치를 바꿨을 때 성능이 조금 향상하였다.

그리고 scheduler를 CosineAnnealingLR을 사용하였다.



위처럼 최고 정확도가 87을 살짝 넘을 때가 있었는데 이때가 87.04%이다.