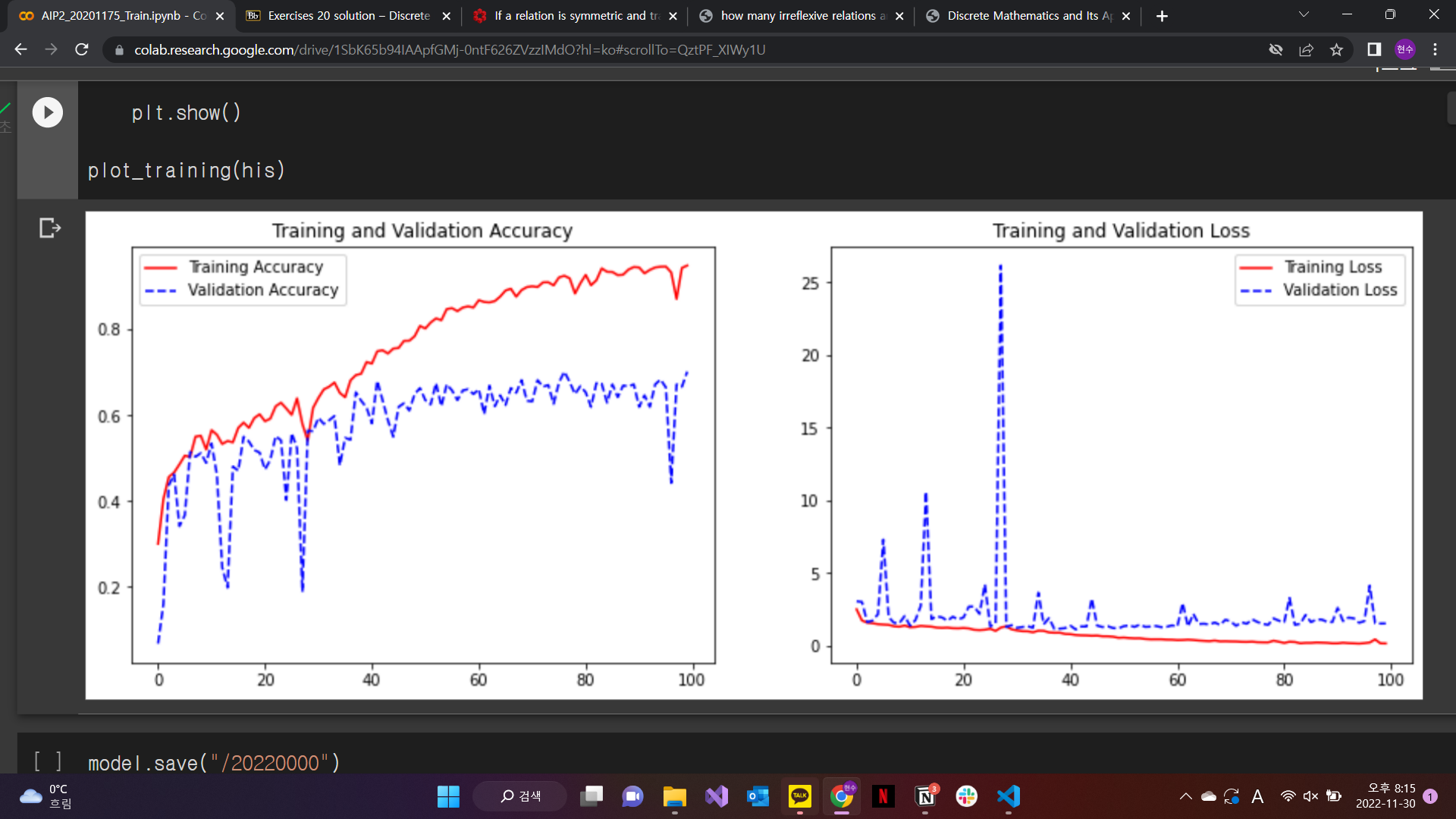
AIP2 Final report – 20201175

1. Model Description

My model is Resnet34. First, I tried the resnet50 model without data augmentation. But there is much overfitting.



(result of resnet50 with epoch 100)

So I think that the model should be more simple. And then I decided my model; (resnet34) + (dropout = 0.5). In my model, regularization is not used. Because there is no significant difference with non-regularization model.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

I used the Resnet34 that has same structure when we learned about resnet34 in lab session 9.

In resnet34, these are layer by layer explanation:

conv2D(64,7, input\_size[200,200,3]) <- my input\_shape is (200,200)

batchnormalization()

Activation(“relu”)

Maxpool2d(3)

Resnet34\_class with filter[64 \*3, 128\*4, 256\*6, 512\*3]

GlobalAvgpool2d()

Flatten()

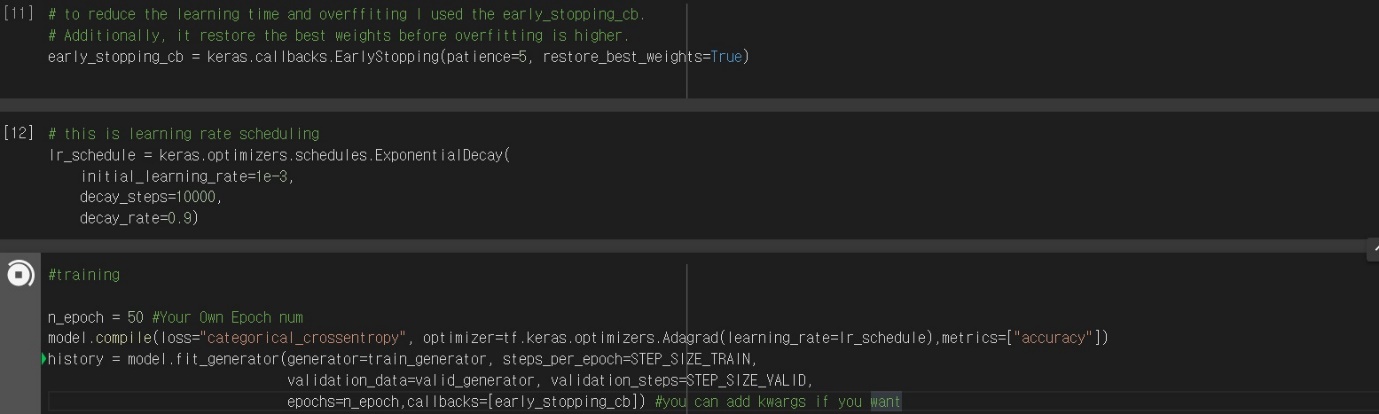
Dropout(0.5) 🡨 very important to reduce overfitting

Dense(n\_classes, activation=”softmax”) <- classify artists

텍스트이(가) 표시된 사진

자동 생성된 설명(there are 21,311,691 parameters)

2. Training (1. optimizer, 2. #of epochs, 3. learning rate scheduling, 4. regularization, 5. data augmentation)

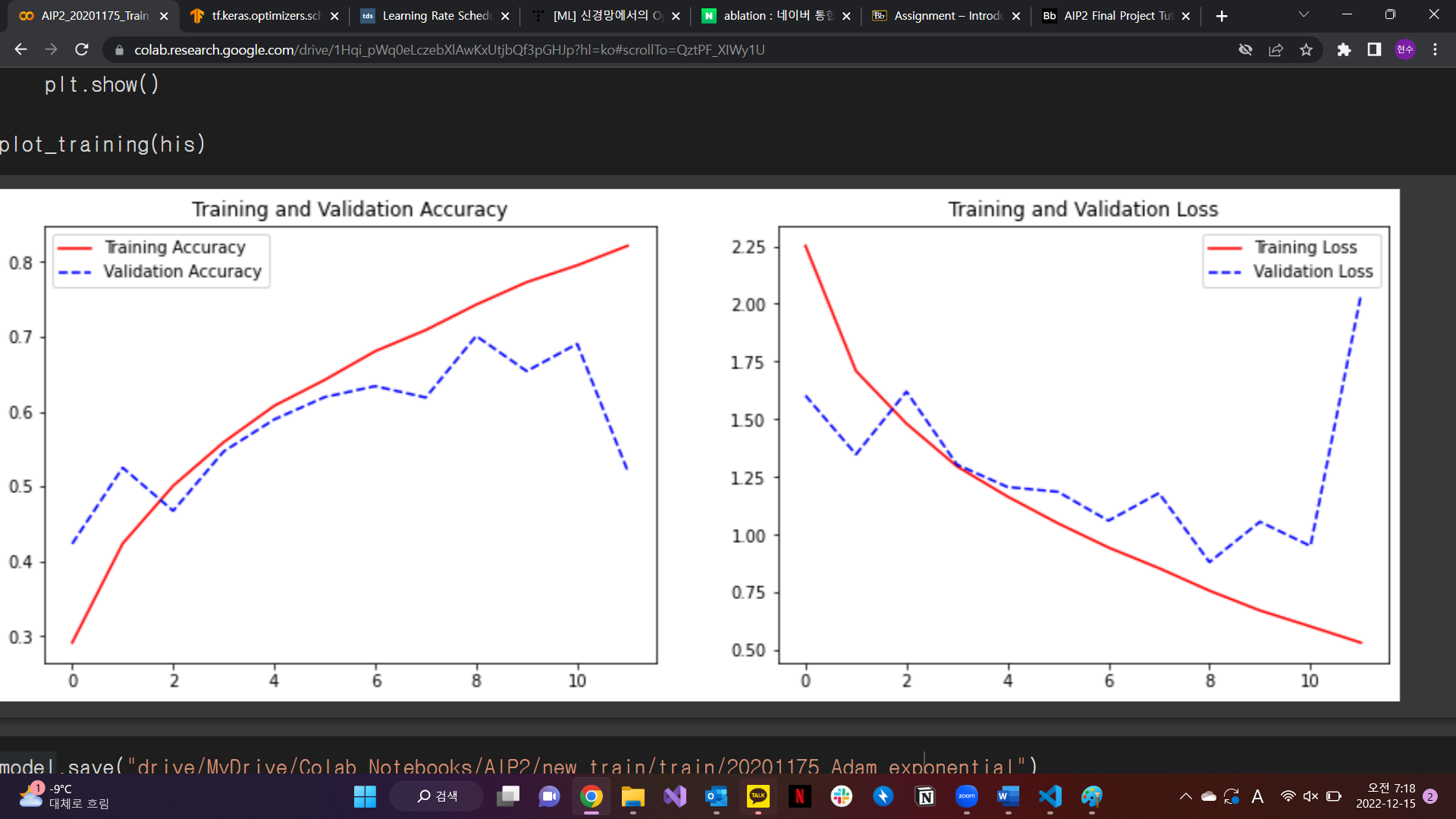


(1) optimizer: “Adam”

In my training, optimizer is “Adam”. That is because when I used “SGD” and “Adagrad” optimizer, the maximum validation\_accuracy is less than accuracy of “Adam”. So, I think that adam is the best optimizer in our project. (nadam don’t have the learning\_rate\_scheduling=exponential. So I except the nadam optimizer)

텍스트, 모니터, 스크린샷이(가) 표시된 사진

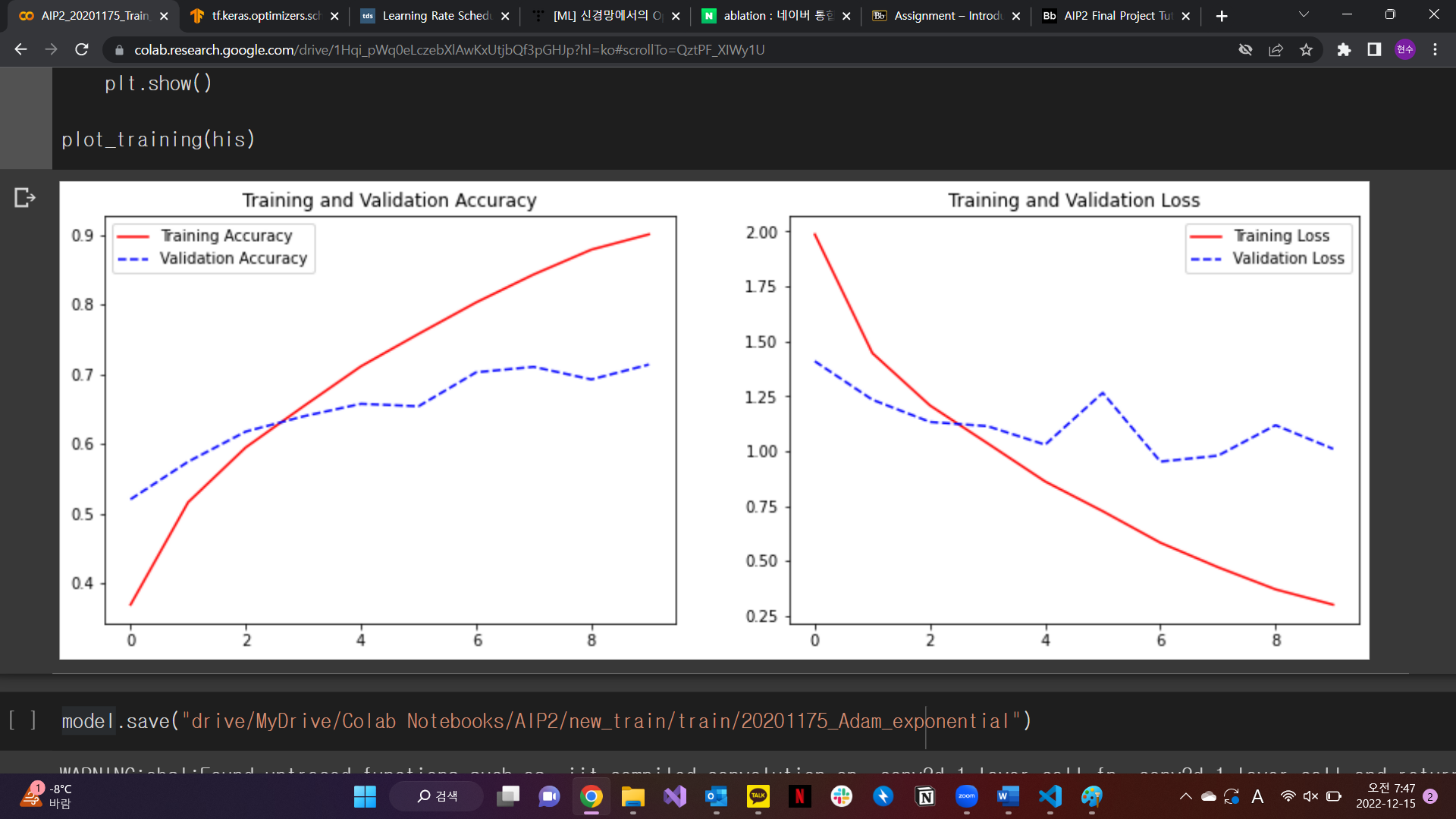
자동 생성된 설명



(result of SGD optimizer)

텍스트이(가) 표시된 사진

자동 생성된 설명



(result of model with Adagrad optimizer)

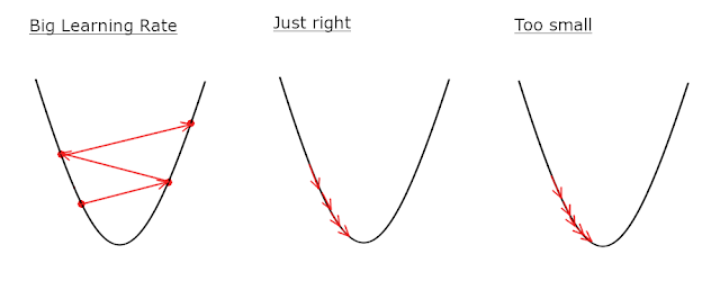
(comparing with the result in performance part, these result(SGD, Adagrad) have lower performances)

(2) # of epochs = 50

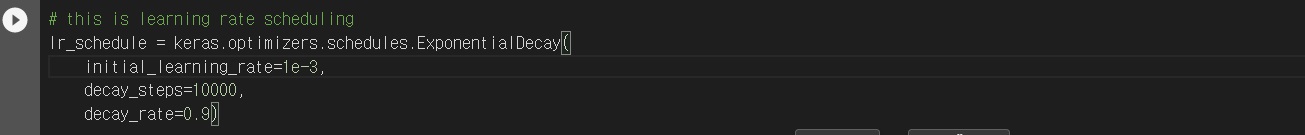
When I first start this project, My training’s # of epochs is 100. But there is overfitting, i.e. its validation accuracy can not exceed some boundary (this boundary is not over 0.7 that seemed to be high accuracy). So I found the way to reduce the overfitting in step of training. That is “early\_stopping”. I can stop when the training goes to overfitting, by using early stopping checkpoint and by using “restore\_best\_weights” attribute, I can restore best weight. Then, I’m done. Thanks to these callbacks, I don’t need to decide appropriate # of epochs. So I just choose the # of epochs some high number (in all trial, early stopping callbacks stopped my training before epochs 50).

(3) learning rate scheduling

Learning rate schedule is important in training. Because when the model is trained, model’s optimizer tries to reduce the loss. In this case, if optimizer’s learning rate is fixed, training can not be well. There are two reasons. First) high fixed learning rate makes the loss lower very fast, but it can not make the loss minimum. Second) low fixed learning rate makes the loss lower slowly (it takes a lot of times).



So, we should reduce the learning rate during training -> learning rate scheduling. In this situation, I tried the exponential scheduling with Adam. (because exponential scheduling is not available at Nadam optimizer). (I used the exponential scheduling because exponential scheduling is evaluated as efficient and well operated scheduler)



Brief statement) At initial\_learning\_rate, the learning rate is multiplied by decay\_rate(=0.9). it means that during training, learning rate will be lower than start. -> good training.

텍스트, 스크린샷, 전자기기, 컴퓨터이(가) 표시된 사진

자동 생성된 설명 (result of model with non-learning rate scheduling, maximum val\_accuracy= 0.82)

(we can know that the accuracy is oscillated at the end of training. -> it means that learning rate scheduling is needed. So I added that on my model)

(4) regularization (L1, L2 regularization)

I don’t use model with regularization. Because there is no significant difference in performance. Even, maximum validation\_accuracy is lower than non-regularization model. So, I skipped it. (it doesn’t work)

텍스트이(가) 표시된 사진

자동 생성된 설명

(model structure with L1 regularization)

텍스트, 컴퓨터, 스크린샷, 모니터이(가) 표시된 사진

자동 생성된 설명

(result with L1 regularization)

텍스트이(가) 표시된 사진

자동 생성된 설명

(model structure with L2 regularization)

텍스트이(가) 표시된 사진

자동 생성된 설명

(result with L2 regularization)

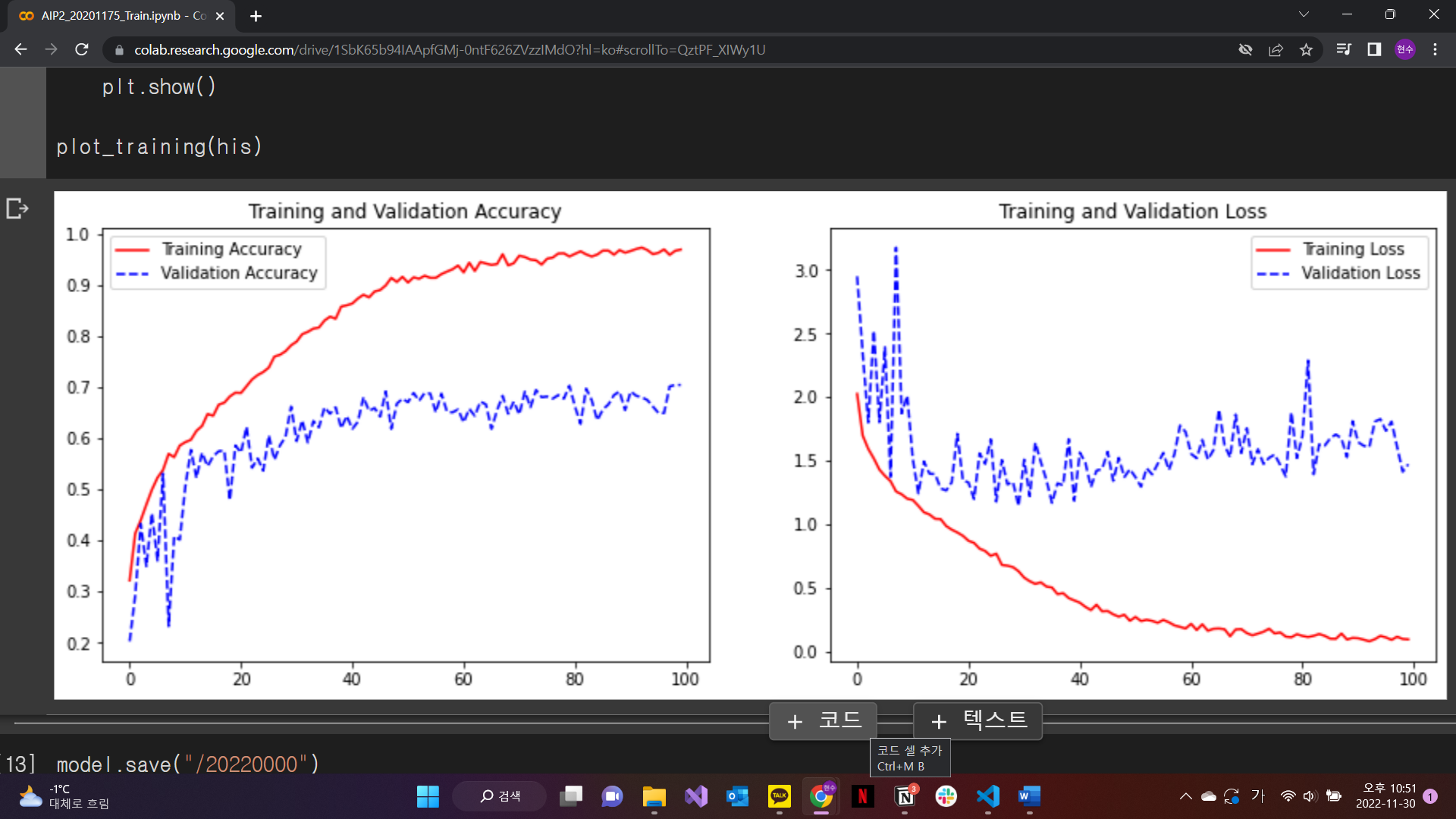
Like this, there is no significant difference with non-regularization model (my result model’s result is shown below). So I skipped to use regularization.

(comparing with the result in performance part, these result(regularization) have lower performances)

(5) data augmentation.

In my final project, data augmentation is very important.

In the original images, they are imbalanced for example, # of images of A artists are about 200 but #of images of B artists are over 800. When I first tried the resnet34 model with [(image=200,200) , (batch\_size= 10), (epoch 100)] there is some overfitting. I think its reason is 1. Data imbalance and 2. Small # of datas.



(result of resnet34+ dropout= 0.5 : with [(image=200,200) , (batch\_size= 10), (epoch 100)]) maximum validation accuracy is 0.66.

When we use imagedatagenerator\_ flow\_from\_directory class, we choose the images and by the attributes of generator, images are chosen. But I think that at that time, because the # of images are imbalanced, they are chosen not equally likely. (My hypothesis)

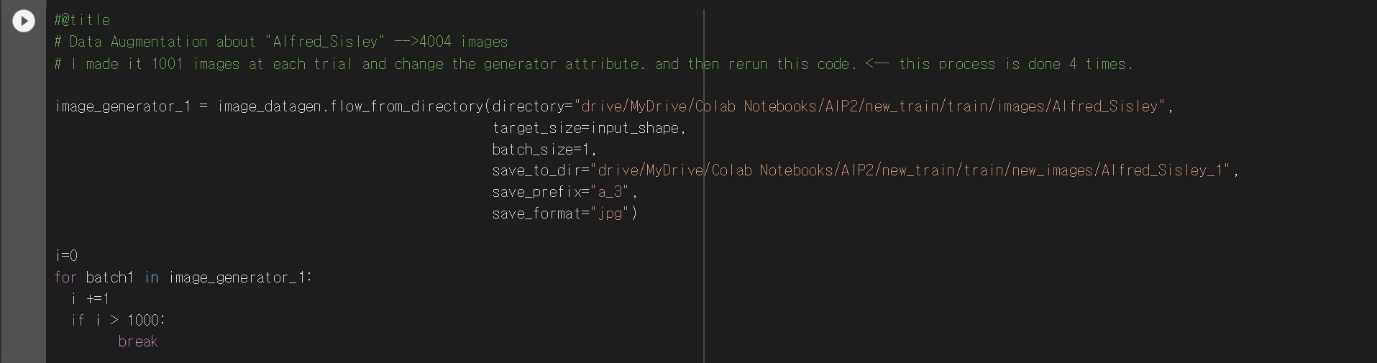
🡪it should be preprocessed. 🡪 oversampling. (make all label have same # of data)

First, I made the 4004 images (about 4000 images) about each artist in my new directory.

Each 1001 images have different attribute of image\_data\_generator (for example, some 1001 images are made by zoom\_range, and other 1001 images are made by rotation.. etc).

텍스트이(가) 표시된 사진

자동 생성된 설명



(make 1001 images)



텍스트, 전자기기, 다른, 여러개이(가) 표시된 사진

자동 생성된 설명

(make 1001 images about one artists with some data augmentation)

텍스트, 실내, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명(1001 images are moved to whole image data directory 🡪 4004 images)

Second, to use these images, we use one more imagedatagenerator. At that time, we just use little attribute about it because we are done about data preprocessing.



Third, by using flow\_from\_directory, we can choose the images equally likely (because image datas are balanced)

텍스트이(가) 표시된 사진

자동 생성된 설명

35,244+8,800= 44,044 = 4004(images) \* 11 (artists)

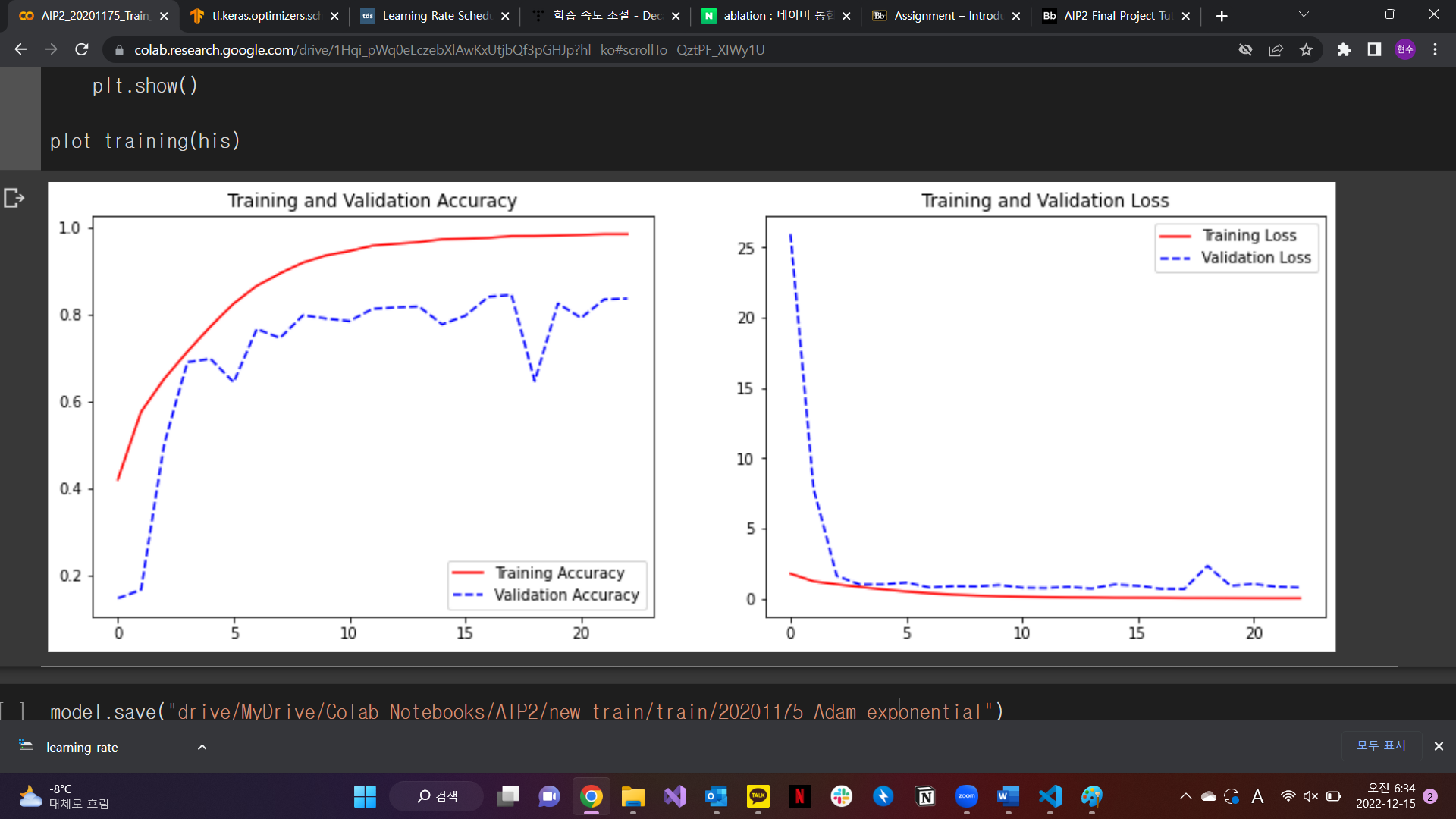
So, it means that imagedata is preprocessed well(oversampling).

3. Performance Evaluation

텍스트, 모니터, 컴퓨터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Thanks to data preprocessing, callbacks and optimizer, I can get accuracy over 0.8. (when I started it, val\_accuracy is lower than 0.6).



This is result of my model. Maximum validation\_accuracy is 0.84. and the overfitting is very low.

This result shows the best accuracy compared to model (with other optimizers, with other model, with other data augmentation)

4. analysis (why choose the model, optimizer, hyperparameter, data augmentation)

I explained most things at the training step. So, I’ll explain remainder (input\_shape, batch\_size).

Input\_shape is (200,200). Because too little shape can’t give the model enough data. And too much shape takes a lot of times when I fit the model (because the # of parameter is increased when shape is increased). So I decided the appropriate input shape (200, 200) not too much, too low.

Batch\_size is “30”. This is high a little about original dataset. Because in original dataset, there are not a lot of images. But after I augment the image data, the # of images is high. So I increased batch\_size from 10 to 30. (when I start this project with no data augmentation, I set batch\_size= 10). So I think 30 is appropriate # of batch\_size.

So my deep learning training is done.

Restatement) Thanks to (dropout, popular model “Resnet34”, data augmentation, learning rate scheduling), I could reduce the overfitting in my training.