**MachineHack Result Report**

Name : ParkJunTae

Rank : 1

First, I am Korean, and there may be a mistake in translation because it is not my native language.  
  
First of all, I was currently attending artificial intelligence graduate school, participating in the competition at the same time as research activities. Especially, I participated in image-related competitions. This is my first time to machinehack hackathon, but I think it was really good to get good results.  
------------------------------------------------------------------------------------------------  
  
Initial Start Method  
  
First, I'd like to explain my way of solving the problem.  
The competition was a matter of classifying whether there was a watermark in the image data.  
Therefore, I hope there is a pretrain model related to this, so

[Watermark-detection](https://github.com/boomb0om/watermark-detection)  
I tried to solve the problem after inputting the image size to the default value of 512x512, referring to the link.

We thought that the larger the image size, the better the watermark could be found, and since the model used the convnext-tiny model, we also wanted to solve the problem by increasing the size of the model.

------------------------------------------------------------------------------------------------  
Introduction to methods  
------------------------------------------------------------------------------------------------  
Image size : 1024 x 1024  
  
Image processing:  
Train - Resize, RandomRotation , Normalize (ImageNet default [0.485, 0.456, 0.406], [0.229, 0.224, 0.225])  
Test - Resize, Normalize (ImageNet default [0.485, 0.456, 0.406], [0.229, 0.224, 0.225])  
  
Train data was used 80% as Train and 20% as Validation data.

------------------------------------------------------------------------------------------------

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

The model is convnext\_large\_in22k, and we used timm to load and use the model.  
Added three fc layers and ReLU(), Dropout() in the middle.

criterion = torch.nn.CrossEntropyLoss()

optimizer = optim.AdamW(params=model.parameters(), lr=0.2e-5)

scheduler = torch.optim.lr\_scheduler.ReduceLROnPlateau(optimizer, mode='max', factor=0.5, patience=2, threshold\_mode='abs', min\_lr=1e-8, verbose=True)

------------------------------------------------------------------------------------------------

Parameter

BATCH\_SIZE = 4

Num epochs = 50

Patience = 6 (early stopping)

------------------------------------------------------------------------------------------------

Finally, the model was put in Watermarks Predictor to return the results.

[Watermark-detection](https://github.com/boomb0om/watermark-detection)

The Watermark Predictor used the link code.