IAA : ENS data challenge

Land cover predictive modeling from satellite images (by Preligens)

ZHAO Yunfei

November 21, 2021

In this report, I am going to show some essential elements of my journey of ENS data challenge during the machine learning course of Mr. Maxime Sangnier. The name of challenge is *Land cover predictive modeling from satellite images*. Given an image from satellite, we need to predict the ratio of each class takes in this image.

Challenge link: https://challengedata.ens.fr/participants/challenges/48/

1 Introduction

In this challenge, given an image of four channels(R-G-B-near-infrared) of size 256×256, we need to assign each pixel to one of ten cover class (histogram of proportion of classes in train dataset is showed in figure 1). And we need to give the proportion of each class in an image as our final output result. So this challenge is a very classical semantic segmentation task. We possess 18491 training samples, to use such a huge amount of data, I found that classical machine learning method like SVM and KNN is not quite adapted. With a small sub training set, these methods give poor performance with long training time. So in the following sections, I am going to present three deep learning models that I have tried and two different loss functions I have designed and a data augmentation methods I choose to use.

2 Data preprocessing

Data analyse I calculate the mean and variance of the pixels value for each channel in training set, and we wanted to standardize the pixels to make them have mean 0 and variance 1. But I think that for a land cover problem like this one, the given data set may generate mean and variance very biased. So we decide to simply divide pixel value by the maximum of pixel value 24356 to normalise it. But this lead to a problem that most of the pixel value close to 0.1 as most of values we have is around 10^3 while, the max value is at 10^4 degree, but this is quite easy to code. (Do you have any advice?)

Data augmentation Data augmentation can help out exploit more information from data and make model have a better generalization. As **Unet** need to have an input image of size 2^n, I haven't added crop operation, but I will add it to DPT model. For now, my data augmentation is a series of flip and rotation, this part is done with Qiaoyu LUO, we found that, one image we can generate 8 different images. An example is showed in figure 2. In pytorch **torch.rot90** is to rotate image from first axe to second axe and is counterclockwise. There respectively, origin image, rotation with -90, 90, 180 degrees, updown flip, updown flip and -90 degrees rotation, updown flip and 90 degrees rotations, updown flip and 180 degrees rotations.

Data separation I randomly select 20% of samples from training set as validation set.

3 Model and Loss selection

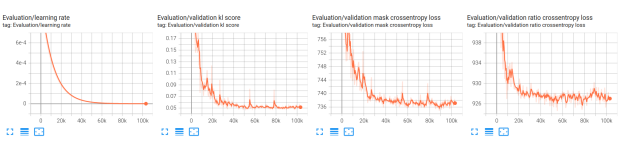
I follow the strategy that during training the model, by selecting the best model with regard to the validation set evaluation result, I restart from the model 10 epochs before and then train the model with whole dataset for 10 epochs. For KNN and SVM, I got a result of KL divergence more than 0.8 (lower than random result provided), so I am going to discuss four deep learning models I have tested. **Loss functions** As mask of an image has far more information than the given proportion label. My loss functions are all based on mask of image with a little bit tuning.

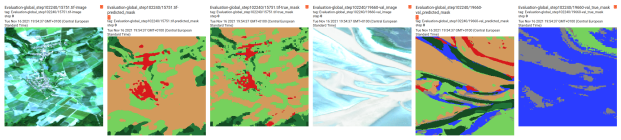
- Cross-entropy loss on predicted mask, the output of an predicted mask is of size $(B, C = 10, h = 256, w = 256)$, we pass it directly to the Pytorh cross-entropy loss (It is composed of softmax $\rightarrow$ log $\rightarrow$ NLLLoss [negative The negative log likelihood loss] for each pixel prediction). During the prediction, we use **Argmax** to select max value indice as the class of a pixel.

- Cross-entropy loss on predicted mask + KL loss on proportion label. As I want to use a softer method to predict the final output, given an output from model, we use softmax on class channel and treat it as the probability of each class, than we sum the probability of pixels along each channel to get a tensor of size $(B, C = 10)$ and I divide it by the sum of these 10 channels to get the final proportion results. Then we add a KL loss between this final output and their given class ratio label. This make a pixel can contribute to several classes based on the predicted probability. During the prediction, we do the same thing as we do for training, instead of selecting class with the maximum probability.

Optimiser I use the Adam optimiser given by the baseline for all models except DTP. For DPT, I used optimiser precised in their paper [1].

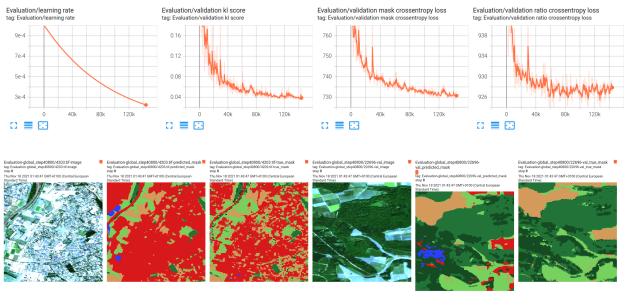
Unet pytorch with one loss





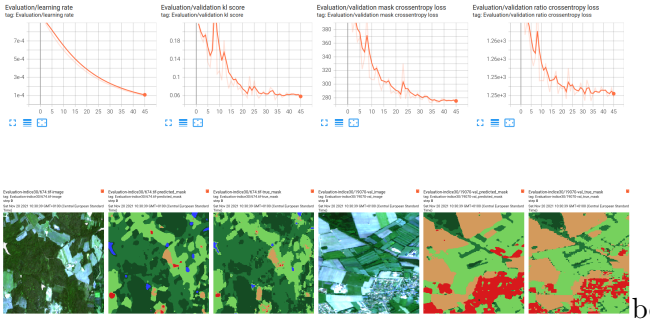
With only the cross entropy loss, it achieve 0.510500495171943 with ealy stop of 56 epochs on test dataset. The loss on training set stop to go down and some predictions on training dataset is not very good. the three images on the left shows an example of an image from validation set, it fails to predict the right mask.

Unet[2] pytorch with two loss



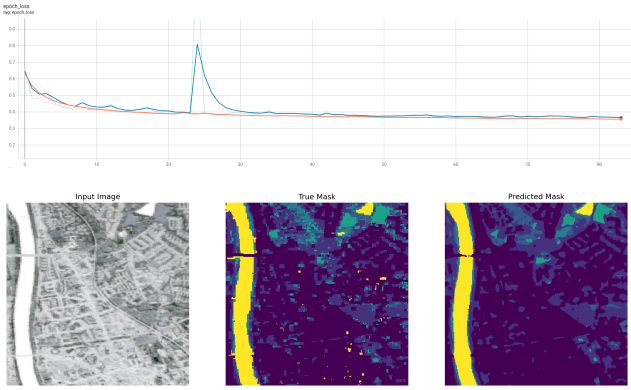
With KL loss and cross entropy loss, it gets a better optimisation result. It achieve 0.452567430783688 on test set.

Big Unet pytorch



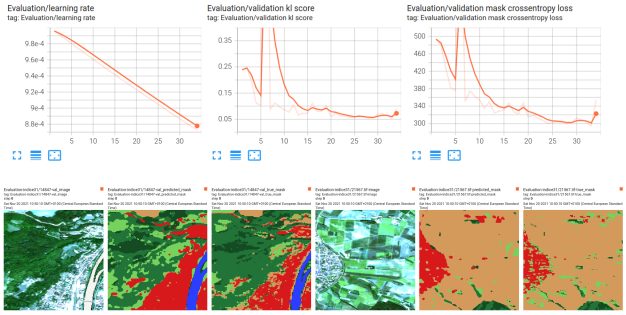
I double the channel dimension for each convolutional layers to solve the under fitting problem. It achieve a better result as 0.229162153376897 on test set.

Unet baseline



The result of baseline shocked me quite a lot. It is implemented on tensorflow with keras, it has exactly the same structure as the structure I use on pytorch, with fewer data augmentation, but the result is miraculously good. It has 0.054093272708472 as final results with 48 epochs. This is a normal Unet and I do not understand why it works so well.

DPT: Vision Transformers for Dense Prediction [1]



Dpt is a model based on transformer. It replace convolutional layer by attention mechanism which is quite popular in computer vision research. It made a new baseline on ADE20K dataset. It use the weights pretained for it's backbone, a resnet50 network by changing the first convolutional layer to adapte our 4 dimensional image. Then I retrain the model on our dataset by changing the decoder output layer to make the head output the class number we need.

It achieves 0.0493445011689579 on test dataset with only the mask loss for training. I notice that this model may need more data for traning as it become overfitting quickly.

bc

4 Impediments and achievements

With a graphic cart on my desktop, the training process is relatively long, but the challenge is quite interesting. With more time, we could try more interesting things. There are some phenomena that I can not explain for now, for example why the same model achieve better results on Tensorflow. There are also things to analyse, for example how optimiser influence our result.

At the time this report was written, I ranked the third on public academic leaderboard. During this experience, I read and understand papars **U-Net**, **Vision Transformers for Dense Prediction** and I get familiar with pytorch and tensorflow frame work.

References

[1] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. *Vision Transformers for Dense Prediction.* 2021. arXiv: 2103.13413 [cs.CV].

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015. arXiv: 1505.04597 [cs.CV].

5 Annex

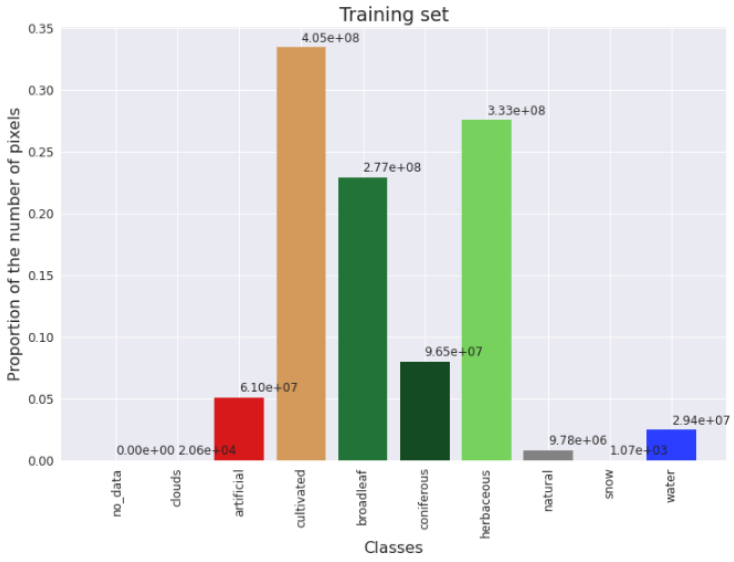


Figure 1: Illustration of class proportion in train dataset

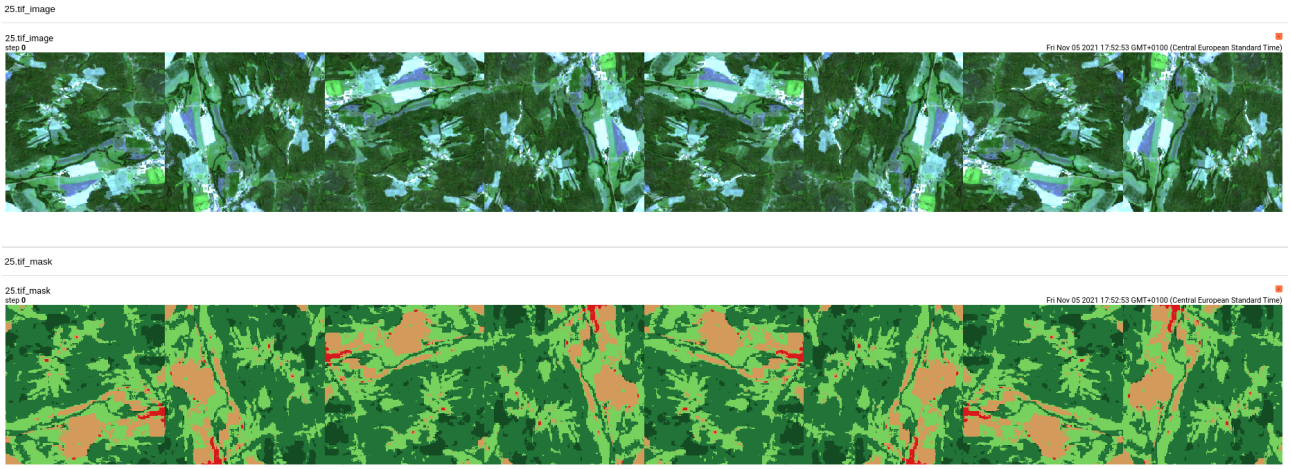


Figure 2: 8 kinds of new image, mask coups generate from one sample from dataset