"At first, I thought it might be more reasonable to split the data by `lesion_id`. But I believed the images won't share the personal information, so it won't be necessary to split on `lesion_id`. I believe splitting the data without consider `lesion_id` should not have data leakage problem."

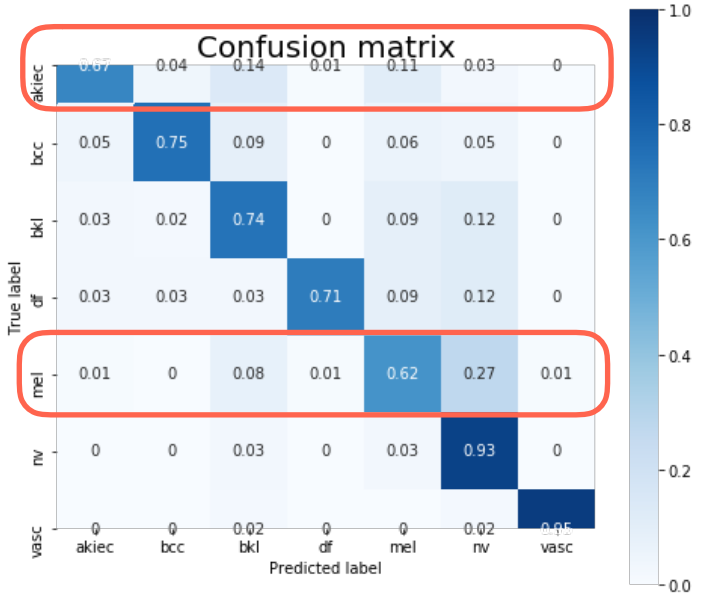
Can you elaborate on this statement? How would you define data leakage and why would it / would it not be affected by splitting on lesion_id?

1. The lesion_id only happened once and won't be duplicated in the future, so we won't have a lesion_id that is both in testing data and training data. Based on this case, I thought splitting the data by lesion_id might be a reasonable scenario. But after I looked into the data, I found the imbalanced issue was very serious. On the other hand, I glanced at several images within the same lesion_id, I couldn't tell whether the images were from the same lesion_id or not. Therefore, I decided to use a stratified split to make sure the model could deal with imbalanced data well, as there's no guarantee that using lesion_id to split data could get a similar distribution on target on validation data.
2. My definition of data leakage: The features contain the information about the target, but that information cannot be collected in the future. Like I mentioned in the previous paragraph, we won't get the same lesion_id in the future. The data leakage only happens if we use lesion_id as a feature to predict skin disease, and we can guess each picture which comes from specific lesion_id. That means, we can get good validation metrics by using the information provided from lesion_id, but in the future, there's no duplicated lesion_id, so we might fail on future testing data.
3. Summary:
    a) Split by lesion_id:
        I. Pros: Same scenario with testing.
        II. Cons: The distribution might be different in training and validation.
    b) Split by stratified strategy:
        I. Pros: Same distribution in training and validation.
        II. Cons: Different scenario with testing (same lesion_id might in both training and validation data).

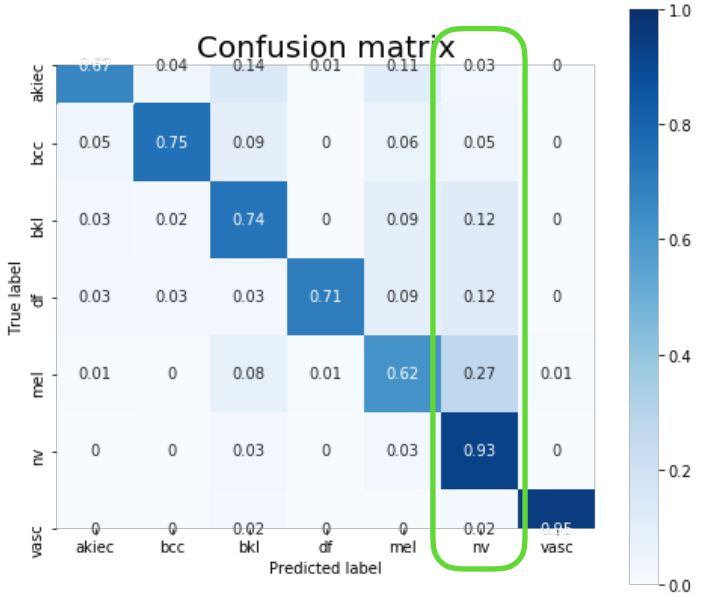
"Implement the weighted cross entropy loss, the weight was decided by the insight from confusion matrix" and ***" Based on the confusion matrix in Version 4, modified the weight of loss function."***

Can you elaborate on how and why you use the confusion matrix for modifying weights of the loss function?

1. At first, I'll list the parameters and the confusion matrix of version 4.
   a) Weights in SE-ResneXt (32x4d) version 4: [0.9, 0.9, 1.0, 1.0, 1.0, 0.6, 0.8]
   b) Weights in SE-ResneXt (32x4d) version 5: [1.0, 0.9, 0.9, 1.0, 1.0, 0.5, 0.7]



2. For the horizontal axis (True label), the two red circled label: ***akiec*** and ***mel***. They got low accuracy (< 70%), so I would like to give them higher weight.
   a) akiec: 0.9 -> 1.0
   b) mel: remained 1.0, but lower other label's weights.



3. For the vertical axis (predicted label), the green circled label ***nv*** has the sum of probabilities are 1.54, which is significantly larger than 1. That means our model was prone to predict the skin disease as Melanocytic Nevi type. So I would like to lower this label's weight.
   a) nv: 0.6 -> 0.5

4. The reason why using confusion matrix is easy to find which classes are over emphasized by model, and which classes should be put more emphasis on.

"Changed Average Precision to Average F1 score. After a long time consideration, I can't tell precision or recall is better in this scenario, so I decided to use F1 score."
It would be interesting to get more info on what were those considerations you thought about when defining what would be a good performance metric for this task.

1. Based on imbalanced data, I thought precision and recall are the most commonly used and easily to explain metrics. Given a certain disease, recall represents the ability of the model to correctly identify those patients with the disease. Precision represents how confident if the patient was identified to have the disease by the model. Although I am not proficient in biostatistics, I have taken a clinical medicine class in graduate school. As far as I know, it's frequently to use sensitivity (recall) as a metric when verifying the efficiency of medicine.
2. In this case, I thought if I use recall as my metrics, this will reflect the situation if the model didn't correctly diagnose those minority classes of disease. On the other hand, precision will reflect on if the model predicts most of the validation data are the same disease. At the begging of this assignment, I thought my model has a serious problem that most of the prediction are in the same class, so I chose precision as validation metric at first.
3. Therefore, based on the point of view I mentioned, I thought both recall and precision are suitable for this case. I believe it will be better to ask a domain export to define which metric will be better. So my final decision is to look after both sides, which is combine recall and precision as F score.

"Removed RandomBrightnessContrast from image augmentation, since most of the images have close brightness. By doing this, also faster the speed of convergance in cross entropy loss."
Can you explain why removing this specific image transformation would result in a speed in convergence of the model?

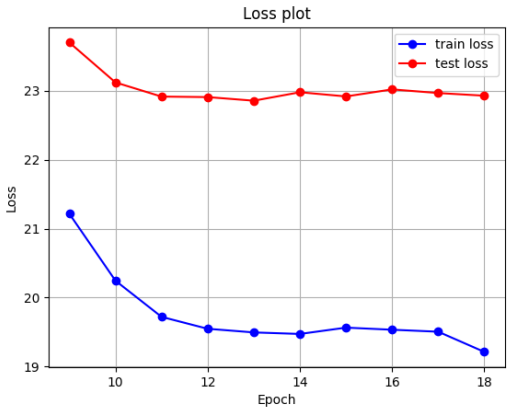
1. I think this was just an observation on the training loss when I compared it with the previous version.
2. To explain this phenomenon, I think there's no theoretical explanation on this question, so I'll just give an intuitive explanation. I added this image transformation at the beginning of this project because it performed well in my experience. Then I found most of the images had similar levels of brightness, so I believe changing the brightness of the image will make the data more diverse but this makes it harder for the model to learn.

"We could found that the validation loss after 5 epochs become unstable in previous version. I believe it's because the learning rate decay is not fast enough"

1. Why do you believe the learning rate decay is the problem when you see this training curve? What is the effect of increasing the learning rate decay? Could there be other reasons for the behaviour you are seeing?

2. Did this make the situation better?

1. Generally, there are two reasons causing training loss or validation loss spike or become unstable:
    a) Learning rate is too large
    b) Batch size is too small
2. In my case, I have tested a proper starting learning rate and batch size in previous neural network architecture (resnet18), therefore, I excluded the possibility of small batch size. So, I believe the reason for unstable validation loss is the decay is not fast enough. There are several methods to solve this problem, like "reduce the learning rate on plateau", "use cosine annealing learning rate", "use cyclic learning rate". Although cyclic learning rate is not aimed at solving this problem, I think the concept is very similar, which is to try to lower the learning rate to get into the local minima.
3. For the second question, I added "cosine annealing learning rate" in version 4, and we can see that the validation loss became stable after version 4. Though I also added transfer learning and edited the weights of cross entropy loss, I couldn't tell how effective the cosine annealing learning rate was. In my experience, I believe increasing the learning rate decay usually helps.

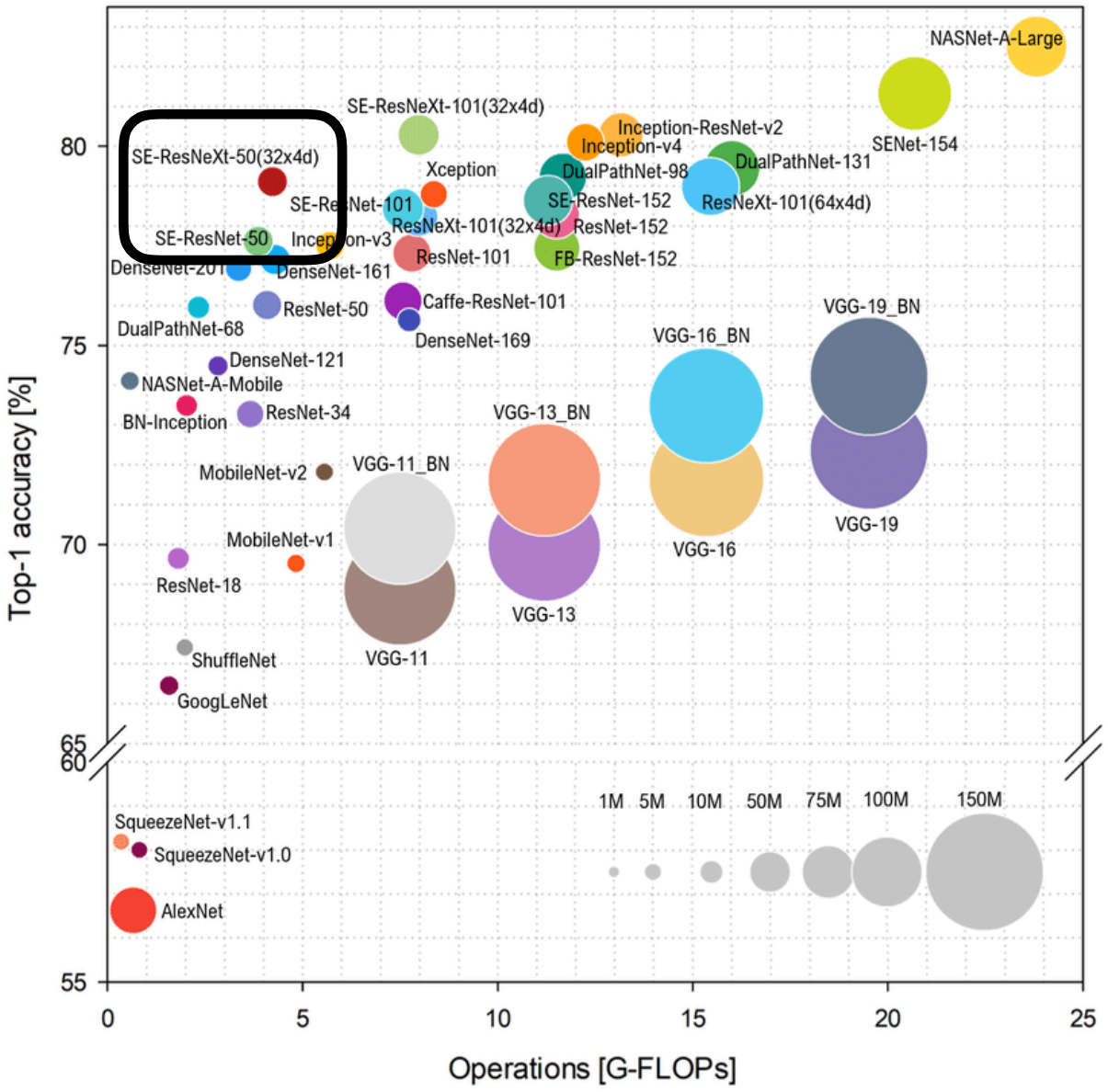


4. I couldn't find the paper I read before, but I found other related papers and topics as reference.
    a) How Does Learning Rate Decay Help Modern Neural Networks?
    b) Cyclical Learning Rates for Training Neural Networks
    c) Kaggle: TGS fluctuation in training loss/metric
    d) Kaggle: TGS how to train/finetune properly for TGS data

In your conclusion you write: ***"The performance of two model (SE-ResneXt50 and SE-Resnet50) in HAM10000 dataset is consist with Imagenet benchmark.".***
Can you elaborate on what you mean by this?

	Weighted Loss	Accuracy	F1 score
SE-ResneXt50	7.169267	0.867044	0.575130
SE-Resnet50	15.252811	0.831291	0.519707

1. I found it's a typo: consist -> consistent. Sorry to cause the misunderstood.
2. The table above is the result I got from validation data. Both two models are based on same parameters (learning rate, batch size, weights of cross entropy, …). We can see the metrics of SE-ResneXt50 are better than the metrics of SE-Resnet50, where SE-ResneXt50 also got better Top-1 accuracy and Top-5 accuracy in Imagenet.



Finally, can you based on your produced results determine which model and settings performs best and motivate why you think so?

1. Based on the average F score and the confusion matrix on the validation data, I would say SE-ResneXt50 (32x4d) performs best with the following parameters:
    a) Learning rate: 0.0001
    b) Batch size: 128
    c) Use version 4 as pretrain
    d) Weighted cross-entropy loss [1.0, 0.9, 0.9, 1.0, 1.0, 0.5, 0.7]
    e) Conduct Test time augmentation 3 times for each validation data.
    f) Oversample the minority classes sample to at least 15% of the majority class sample.