**CSE3001 Assignment Report**

This report chronicles answers to questions raised while completing the CSE3001 assignment.

# Task 1

## Task 1a

### Data issues

What did you check for?

* I examined the distribution of the training, validation dataset across different classes.

Data Issues Present:

* Class Imbalance: The most prominent issue observed in this dataset is the significant class imbalance. The class NV has a markedly higher number of samples compared to all other classes. Such a distribution can lead to a model that is biased towards predictions for the NV class since it provides more examples of this class during training. This can affect the model's performance on underrepresented classes, potentially leading to poorer recall or accuracy for those classes.
* Sparse Representation: Some classes, such as MEL, BCC, and especially DF and VASC, have a very limited number of examples. Sparse representation can be problematic because the model might not see a diverse set of examples for these classes, which can hinder its ability to generalize well to unseen data for these classes.
* Potential Overfitting: Given the imbalance, the model might overfit to the NV class due to its overwhelming presence in the dataset. It might start recognizing patterns specific to the training data for this class that don't generalize well to new, unseen data.

## Task 1b

### Why not use random\_split?

* In the context of this dataset, the distribution of classes is imbalanced as evident from the provided distribution graph. Using random\_split would randomly split the dataset into training and validation sets without considering the distribution of classes. This might lead to scenarios where certain underrepresented classes might have very few or even no samples in the validation set, making it challenging to evaluate the model's performance on those classes.
* To ensure that each class is adequately represented in both the training and validation sets, it's preferable to use a stratified splitting approach. Stratified splitting maintains the original distribution of classes in both subsets, ensuring a more representative split and a better evaluation of the model's performance across all classes.

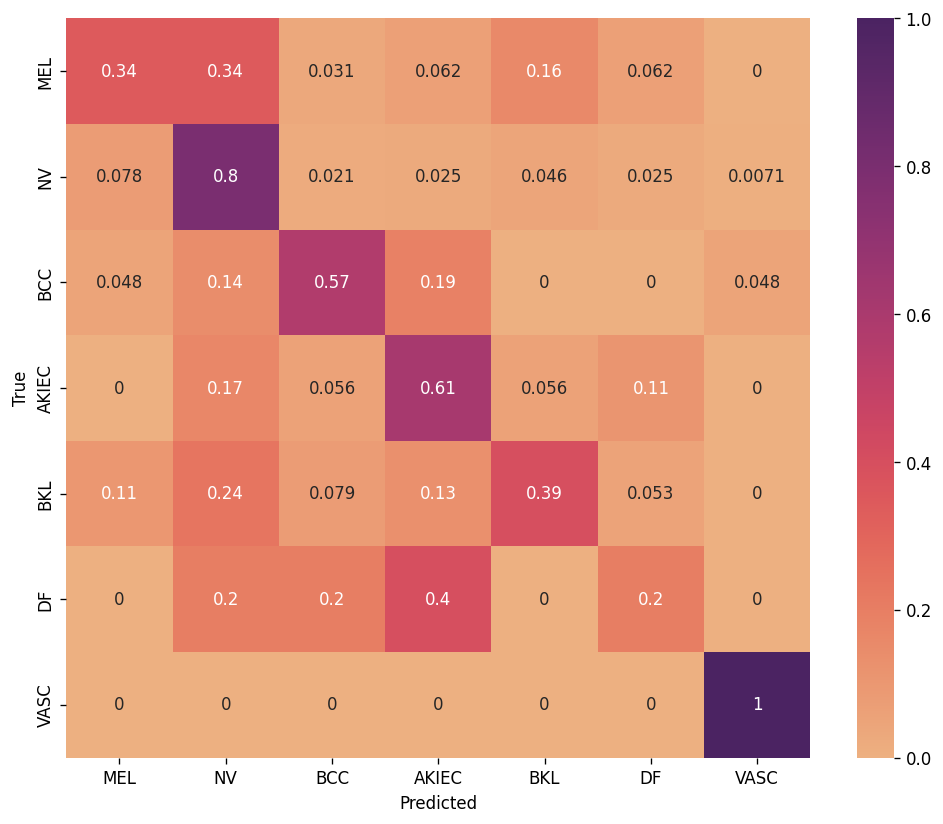
## Task 1c

### Reduce epoch time

Two ways to significantly reduce how long each epoch takes for debugging purposes:

* Use a Subset of the Data: Instead of using the entire dataset, select a small random subset of the data for debugging purposes. This provides the benefit of running the training loop faster while still using real data.
* Reduce the Model Complexity: Temporarily use a simpler and smaller model architecture. This ensures that the forward and backward passes are faster. Once the debugging is done, you can revert to the original, more complex model.

### Confusion matrix



Performance Metrics:

* Training Accuracy: 65.841%
* Validation Accuracy: 55.974%
* Training Loss: 0.94671
* Validation Loss: 0.84478
* Training UAR: 65.841%
* Validation UAR: 55.974%

Confusion Matrix Analysis:

* MEL class: The model correctly identifies this class 34% of the time. However, it often confuses it with the NV class (34% of times).
* NV class: The model performs relatively well on this class with a correct prediction rate of 80%. However, it still confuses it with the MEL and BKL classes.
* BCC class: The class is correctly identified 57% of the time but often gets misclassified as NV or AKIEC.
* AKIEC class: Correctly identified 61% of the time but has confusion with NV and BKL classes.
* BKL class: This class has the most distributed confusion with other classes. It's correctly identified only 39% of the time.
* DF class: The model seems to struggle with this class, with a 40% correct prediction rate, but a 20% confusion rate each with NV and BKL classes.
* VASC class: The model performs exceptionally well on this class with 100% accuracy. However, it's essential to consider the number of instances of this class in the dataset, as high performance on classes with fewer instances may not necessarily indicate good generalizability.

General Observations:

* The model's performance on the validation set is lower than the training set across all metrics, indicating potential overfitting.
* The UAR (Unweighted Average Recall) being equal to the accuracy suggests that the dataset might be somewhat balanced, or the model's performance across classes is relatively uniform.
* The model seems to have a specific challenge distinguishing between some classes like MEL and NV, indicating that these classes might be closely related or have similar features in the dataset.

## Task 1d

### Account for data issues

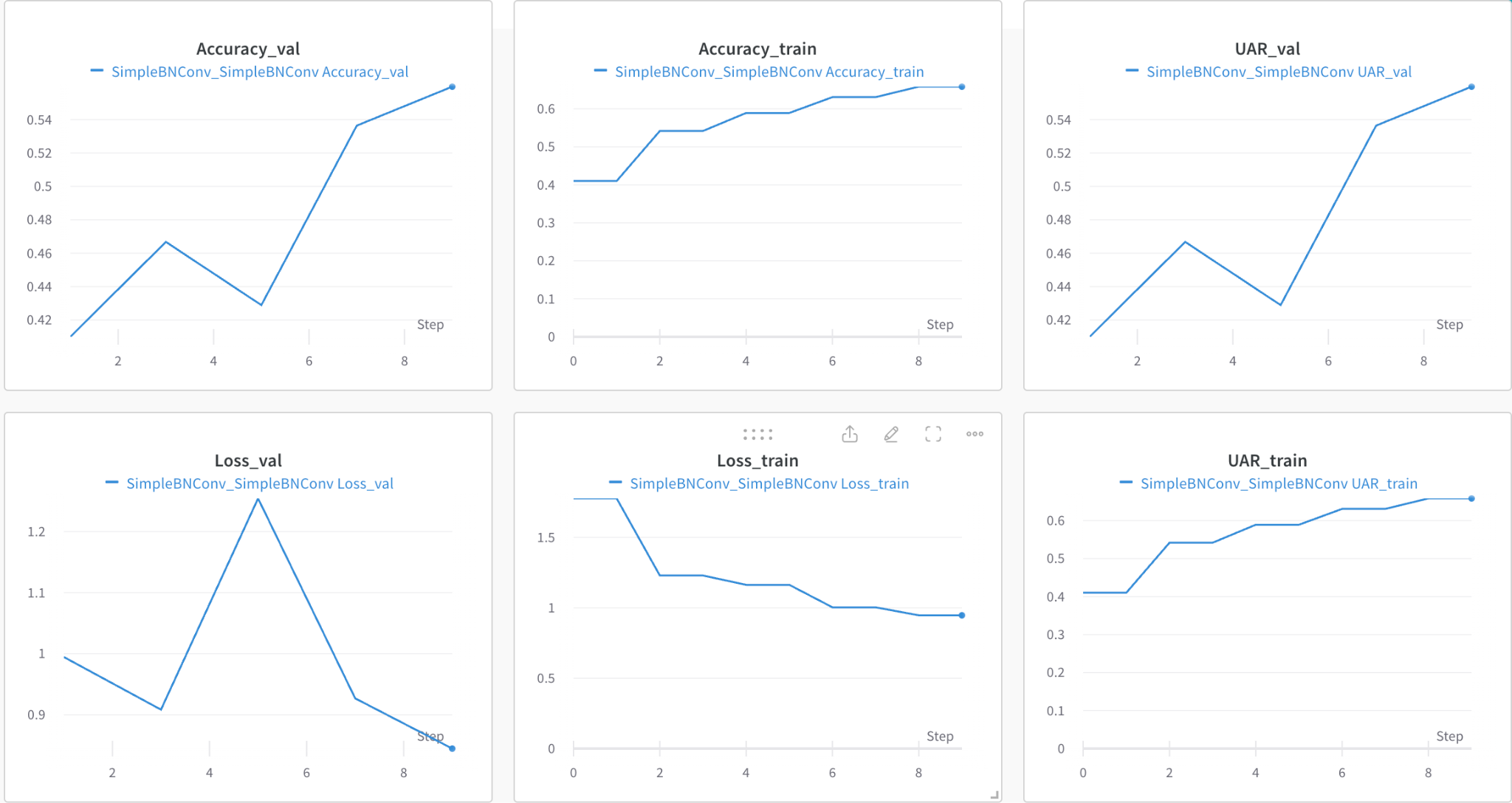
* Handling Missing Values: The provided code starts by dropping rows with missing values from both the training and validation dataframes. This is an effective, straightforward method to ensure the model isn't trained on incomplete data, which can lead to misleading or unpredictable results.
* Class-aware Sampling with WeightedRandomSampler:
* The code calculates the count of each class in the training labels.
* Based on the class counts, it then computes weights for each class. The weight of a class is inversely proportional to its frequency. This means that underrepresented classes will have a higher weight.
* Sample weights are assigned based on the class of each sample.
* Finally, the WeightedRandomSampler is used with the sample weights, ensuring a balanced sampling of classes during training.
* Device Handling: The code is structured to check if CUDA is available, which will allow it to use GPU for faster training. This ensures that you utilise the best computational resources available, leading to efficient model training.
* Class-aware Sampling: While the data loader itself isn't provided, it's vital to ensure that there's balanced class sampling. If there's a class imbalance in the dataset, one should consider using techniques such as oversampling, undersampling, or synthetic data generation.
* Metrics: The code uses macro-averaged accuracy and UAR (Unweighted Average Recall) to evaluate the model. These metrics are particularly useful for multi-class classification problems with imbalanced datasets because they give equal weight to each class irrespective of their sample size.
* Early Stopping: The training loop has a condition to stop training early if the validation accuracy and UAR fall within a specific range. This is beneficial to prevent overfitting and save computational resources if the model has already reached an acceptable level of performance.

## Task 1e

### Vertical Flips

* Nature of Data: The dataset is for lesions. Lesions typically have orientations, and flipping them vertically might create unrealistic representations. In medical imaging, orientation can be crucial. A lesion that appears on the top of an organ or tissue might have a different implication than one on the bottom. Adding vertical flips could introduce confusion.
* Consistency with Reality: While horizontal flips might still be plausible (since a lesion on the left or right might still look natural), vertical flips can produce unrealistic images that the model might never encounter in real-world scenarios.
* Conclusion: Given the potential issues, random vertical flips might not be appropriate for this dataset. However, the final decision would require a deeper understanding of the domain and consultation with medical experts.

### Effect of Augmentation

* Random Horizontal Flip: This can introduce variability without creating unrealistic images.
* Random Rotation (±5°): This is a minor adjustment which might help the model generalize to slightly tilted images.
* Color Jitter: The parameters have been reduced, ensuring that the color variations are subtle and the augmented images still look realistic.
* Random Resized Crop: This focuses on different parts of the image with adjusted scales and ratios. It can help the model recognize lesions even if they aren't centered or if they appear at various scales.  
  

## 

## Task 1f

### Experiments

* <See attached excel document>
* <See attached Weights and Bias Report>
* Write a discussion about the key findings from the experimental results.

…

| **Name** | **State** | **Notes** | **User** | **Tags** | **Created** | **Runtime** | **Sweep** | **batch\_size** | **epochs** | **learning\_rate** | **Accuracy\_train** | **Accuracy\_val** | **Loss\_train** | **Loss\_val** | **UAR\_train** | **UAR\_val** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SimpleBNConvModified\_SimpleBNConvModified, lr = 0.0001** | finished | - | minhhung171002 |  | 2023-10-31T10:50:02.000Z | 532 |  | 64 | 5 | 0.0001 | 0.5673296451568600 | 0.5596177577972410 | 1.1394743957418100 | 0.9880775383540560 | 0.5673296451568600 | 0.5596177577972410 |
| **SimpleBNConv, lr =0,0001** | finished | - | minhhung171002 |  | 2023-10-31T10:25:06.000Z | 534 |  | 64 | 5 | 0.0001 | 0.6248970031738280 | 0.5390338897705080 | 1.050576519458850 | 1.0340627006122000 | 0.6248970031738280 | 0.5390338897705080 |
| **SimpleBNConv, lr = 0,01** | finished | - | minhhung171002 |  | 2023-10-31T10:13:34.000Z | 540 |  | 64 | 5 | 0.01 | 0.40580010414123500 | 0.42818742990493800 | 1.5470296245940200 | 1.5972655160086500 | 0.40580010414123500 | 0.42818742990493800 |
| **ResNet50Model\_ResNet50Model, SoftMarginLoss()** | finished | - | minhhung171002 |  | 2023-10-31T09:47:49.000Z | 547 |  | 64 | 5 | 0.01 | 0.12816335260868100 | 0.12030744552612300 | 0.5941261649131780 | 0.5941261649131780 | 0.12816335260868100 | 0.12030744552612300 |
| **ResNet50Model\_ResNet50Model** | finished | - | minhhung171002 |  | 2023-10-31T09:33:29.000Z | 547 |  | 64 | 5 | 0.001 | 0.6868139505386350 | 0.6231076121330260 | 0.9111488233221340 | 0.9910100272723610 | 0.6868139505386350 | 0.6231076121330260 |
| **SimpleBNConvModified\_SimpleBNConv** | finished | - | minhhung171002 |  | 2023-10-31T07:56:04.000Z | 1153 |  | 64 | 5 | 0.001 | 0.6506491899490360 | 0.42353591322898900 | 0.9463043606027640 | 1.5756194591522200 | 0.6506491899490360 | 0.42353591322898900 |
| **SimpleBNConv\_SimpleBNConv** | finished | - | minhhung171002 |  | 2023-10-31T07:31:16.000Z | 1120 |  | 64 | 5 | 0.001 | 0.6584146022796630 | 0.5597399473190310 | 0.9467067452187230 | 0.8447821480887280 | 0.6584146022796630 | 0.5597399473190310 |
| **ResNet18Model\_Resnet18** | finished | - | minhhung171002 |  | 2023-10-31T06:37:57.000Z | 2770 |  | 64 | 5 | 0.001 | 0.6641706824302670 | 0.6206556558609010 | 0.9531731592847950 | 0.9948338525635860 | 0.6641706824302670 | 0.6206556558609010 |

Model Architectures and Configurations:

I experimented with the following model architectures:

* SimpleBNConvModified
* Variations: Two variations, one with a learning rate (lr) of 0.0001 and another without specific lr notes.
* SimpleBNConv
* Variations: Two configurations with learning rates of 0.0001 and 0.01, respectively.
* ResNet50Model
* Variations: Two variations, one with SoftMarginLoss and another generic ResNet50 model without specific configurations.
* ResNet18Model
* Variation: A basic ResNet18 architecture.

Here are some general insights and discussions based on the provided architectures:

* Learning Rates: The learning rate is a crucial hyperparameter. A low learning rate (like 0.0001) might lead to slow convergence but can be more accurate, while a higher rate (like 0.01) might lead to faster training but risks overshooting the optimal point.
* Model Complexity: Simple architectures like SimpleBNConv and its modified version are likely quicker to train and might prevent overfitting on smaller datasets. In contrast, complex models like ResNet50 and ResNet18 might capture more intricate patterns but require more data and might be prone to overfitting without proper regularization.
* Loss Function: The ResNet50 model variation with SoftMarginLoss suggests experimentation with different loss functions. SoftMarginLoss, a type of hinge loss, might make the model robust against misclassified examples, especially in binary classification tasks.

### [Challenge] Batch size

Why can this be good in terms of gradients?

* Frequent Updates: Updating weights more frequently can lead to faster convergence as the model is refining its understanding of the data more regularly.
* Noise in Gradients: Smaller batch sizes introduce noise in the gradient estimates, which can have a regularizing effect and lower the generalization error. This noise can also help escape shallow local minima.
* Memory Efficiency: Smaller batches require less memory, allowing you to train deeper models or use larger input sizes.

However, there are caveats:

* If the batch size is too small, the noise in the gradient can be too high, leading to unstable training and the need for a lower learning rate.
* Each update, especially on hardware accelerators like GPUs, has an overhead. Too many updates can make training inefficient.