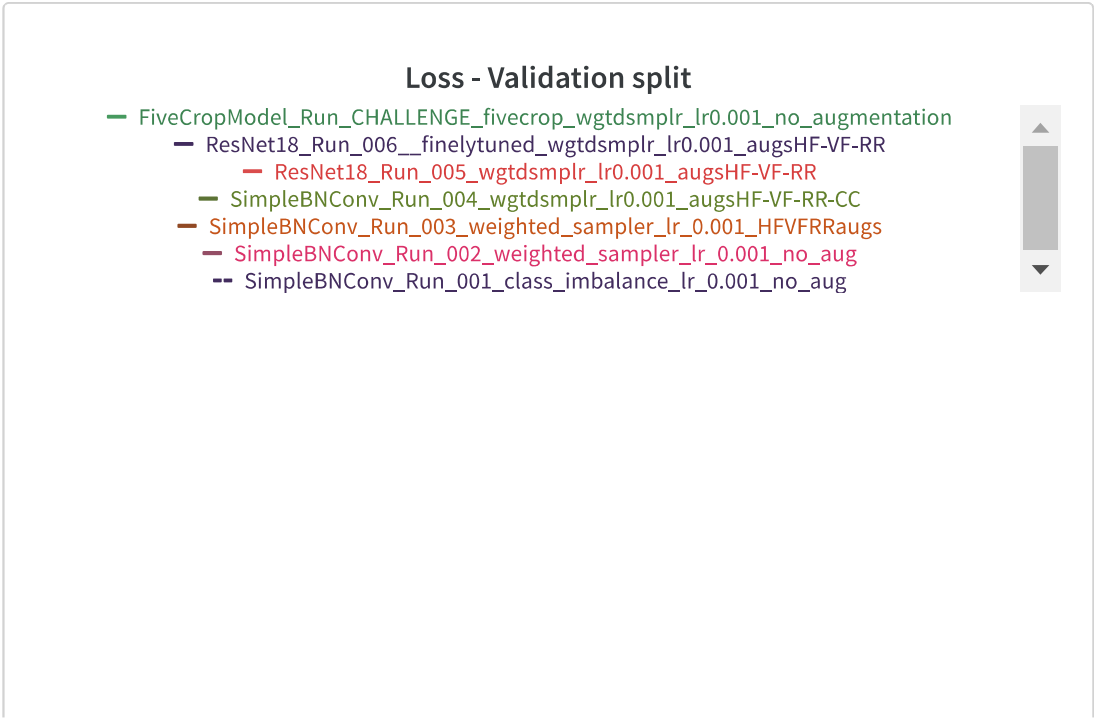
Skin lesion classification task

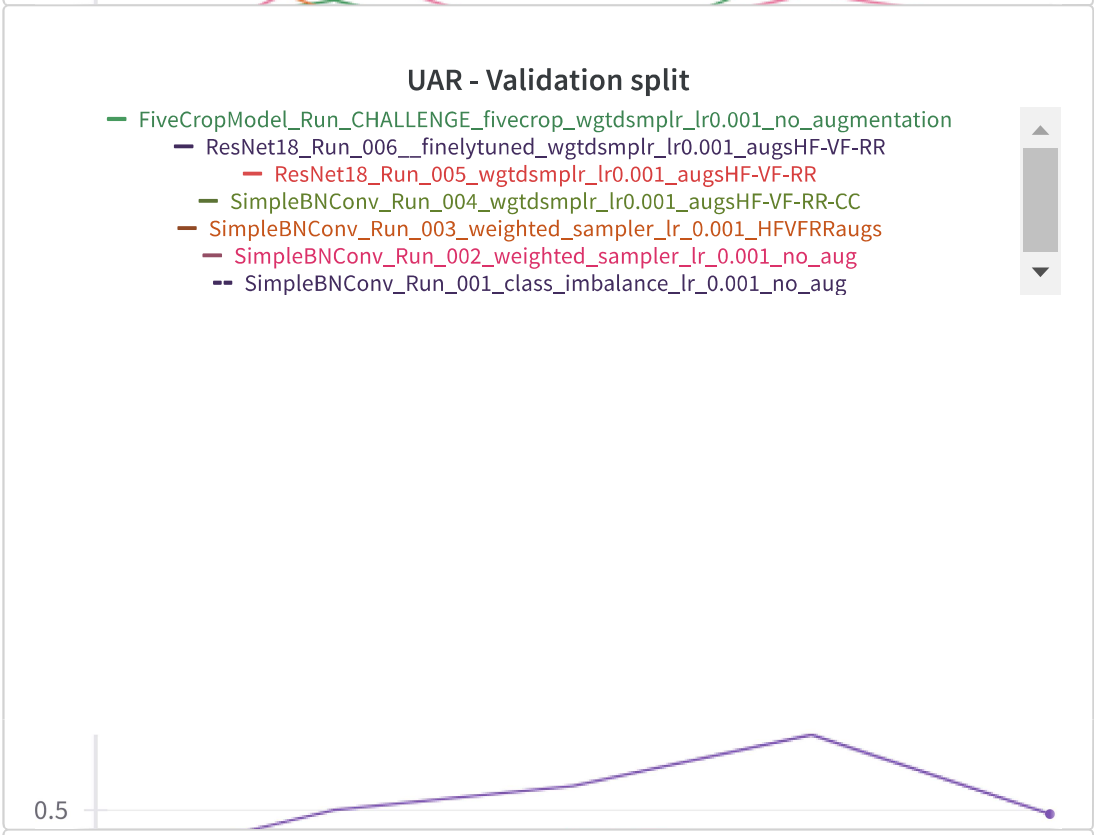
Performance of neural network models with various hyperparameters

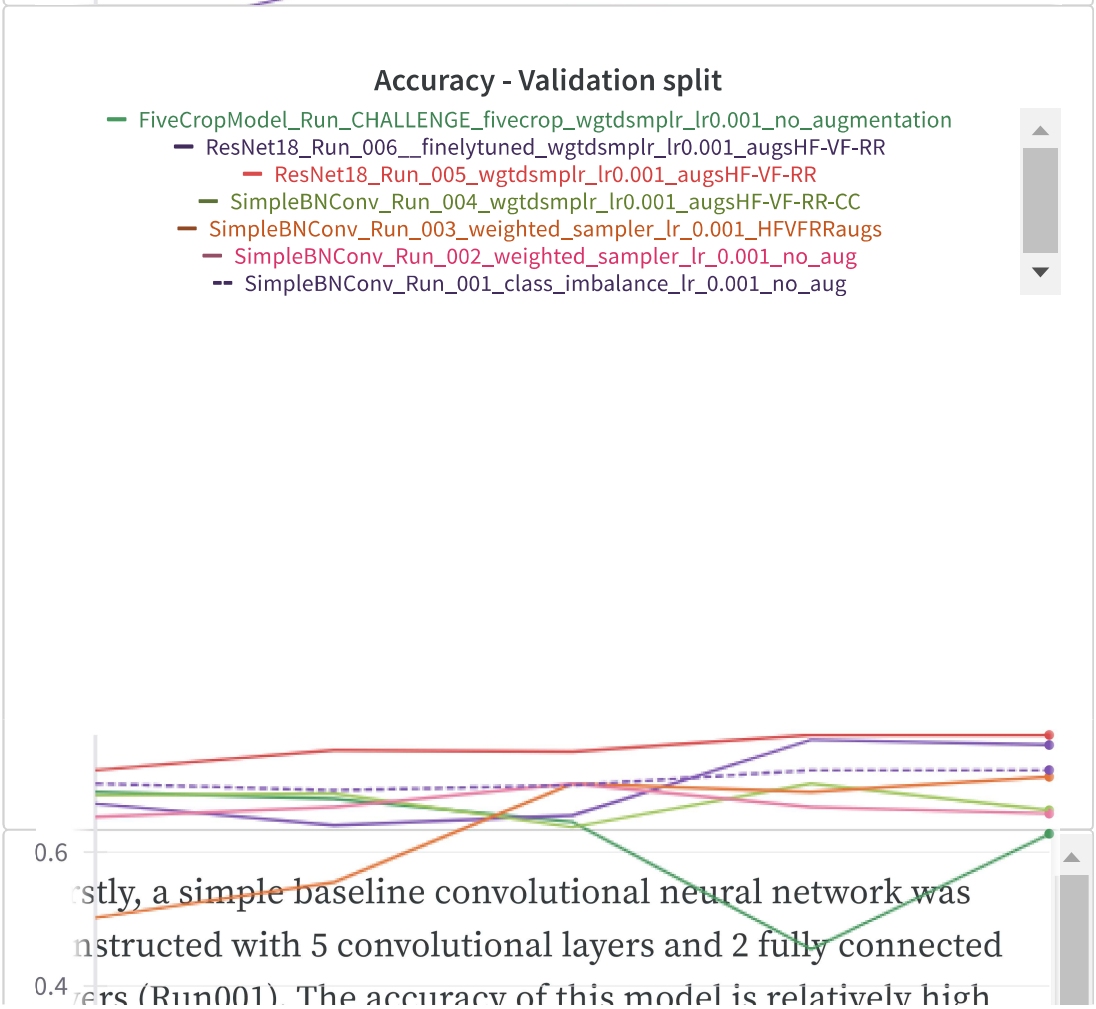
Matthew Finster

PART 1: Establishing a baseline model

Custom Convolutional neural network vs ResNet18 pre-trained model







stly, a simple baseline convolutional neural network was
nstructed with 5 convolutional layers and 2 fully connected
ers (Run001). The accuracy of this model is relatively high

7225) however there was a large class imbalance causing a low
R (0.2813) on the validation set.

Run002, a weighted sampler was introduced which decreased accuracy (0.6575) yet significantly improved UAR (0.4669) on the validation set.

In Run003, augmentations (horizontal flips, vertical flips, random rotation) were applied on the training set. Accuracy

increased (0.7125) but there was a significant decrease in UAR (0.4371) on the validation set.

In Run004, centre cropping was also applied to all images on top of the existing augmentations in previous run. This led to further decreases in accuracy (0.6625) and UAR (0.4281) on the validation set.

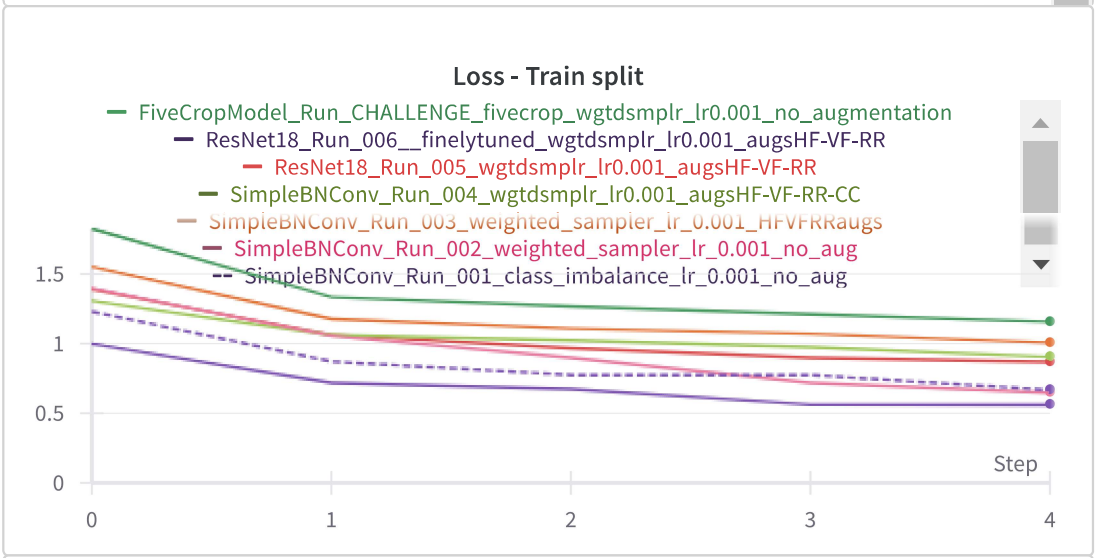
In Run005, a ResNet18 pre-trained model with 20 convolutional layers and 1 fully connected layer was constructed and used. All weights were frozen except for the final fully connected layer in this model. The same augmentations were used (without centre cropping as this resulted in decreased performance). The accuracy increased using this model (0.775) however the UAR decreased again (0.4264) on the validation set.

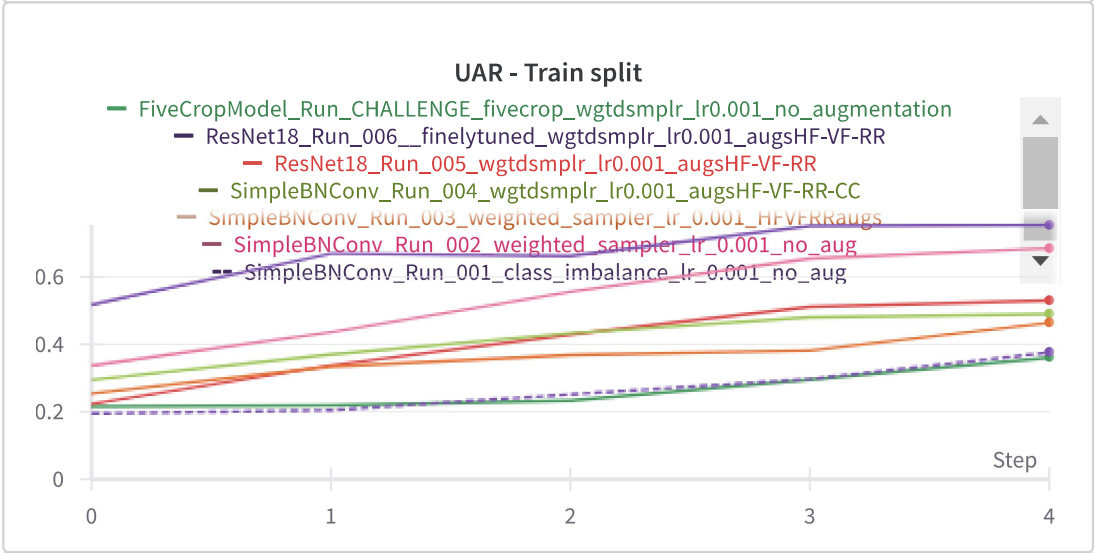
In Run006, the ResNet18 model's weights were unfrozen and the model was tested again. The accuracy remained high (0.76) while the UAR increased to its highest level yet (0.4956). Both ResNet models also resulted in the best loss performance compared to the custom built convolutional neural network with only 5 convolutional layers.

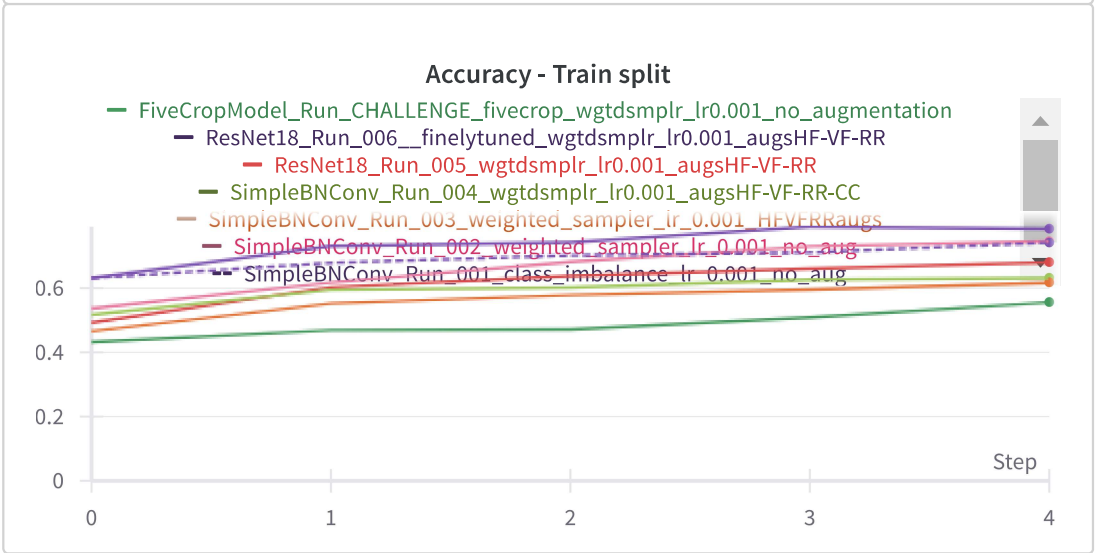
For comparison, another model was contructed that uses 5 crops at once. For this model, the other augmentations were removed although the weighted sampler was included. This resulted in signficantly worse performance in both accuracy (0.6275) and

UAR (0.2914) on the validation set.

Given these results, the model from Run006 was chosen to continue for comparison. Accuracy, UAR and Loss from the Training split are also included below in graphs for reference.

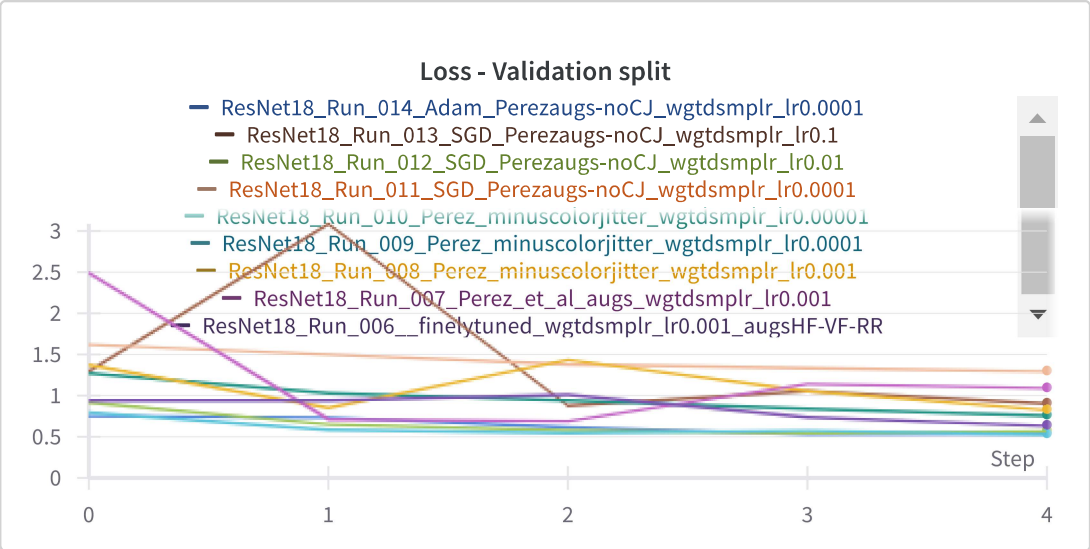


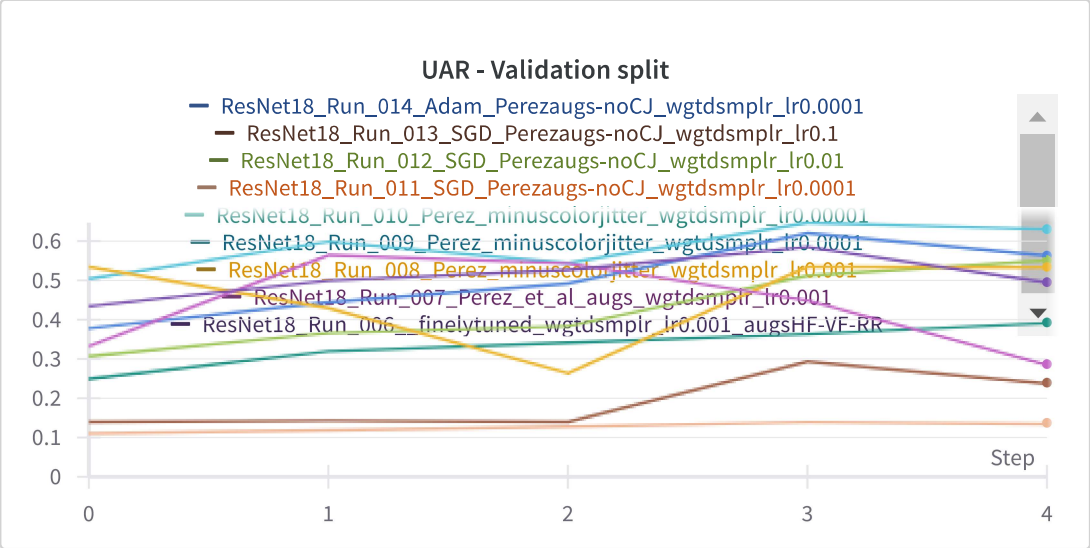


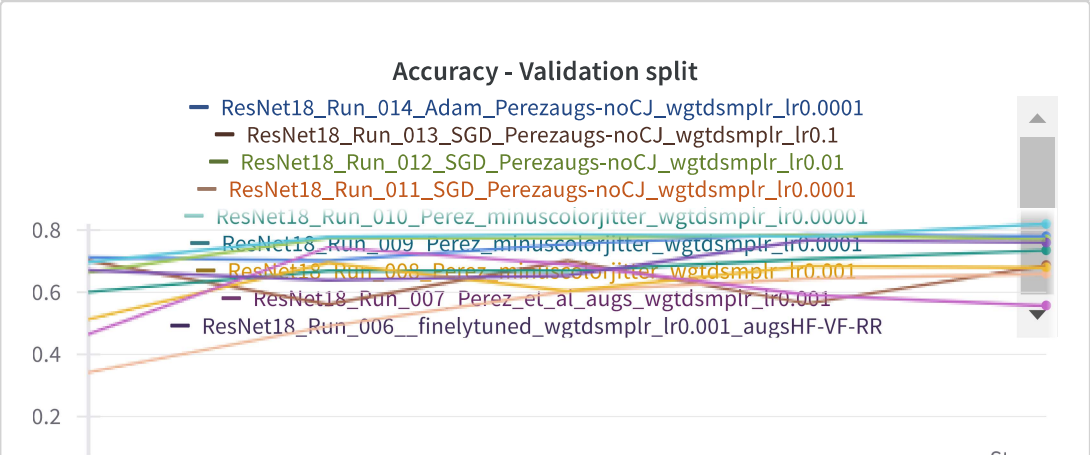


PART 2: Improving ResNet18 performance

Changing augmentations, learning rates and optimisers







Note, Run006 was kept in these graphs to use for comparison.

Prior to Run007, 'Data Augmentation for Skin Lesion Analysis' by Perez, Vosconcelos, Avila & Valle (2018) was read. These authors found the most optimal augmentation techniques (Random cropping, Affine, Flips, and Saturation&Contrast&Hue). The previous augmentations were removed from the model. Affine was applied before flipping and ColorJitter was applied after flipping. These were combined using Random Apply with a probability of 0.5. Compared to Run006, these augmentations resulted in decreased accuracy (0.5575 compared to 0.76) and significantly worse UAR (0.2869 compared to 0.4956). It was then theorised that perhaps the natural colours of the skin lesions were actually important for training the model.

In Run008, ColorJitter was removed as an augmentation but the other augmentations recommended by Perez, et al. (2018) were maintained in the model. This resulted in significantly improved performance in accuracy (0.68) and UAR (0.5345) on the validation set. This was the highest UAR achieved up until this point. Given the combination of augmentations from the previous step led to the best performance so far, it was expected that using the same model with slightly different hyperparameters may lead to improved results, starting with the learning rate.

In Run009, the learning rate was decreased to 0.0001 so that the model would update the weights by smaller amounts after each

batch. This resulted in again the best performance thus far with an accuracy of 0.82 and a UAR of 0.6307 on the validation set.

In Run010, the learning rate was decreased again, but this caused the model to learn too slowly, resulting in decreased performance on the validation set

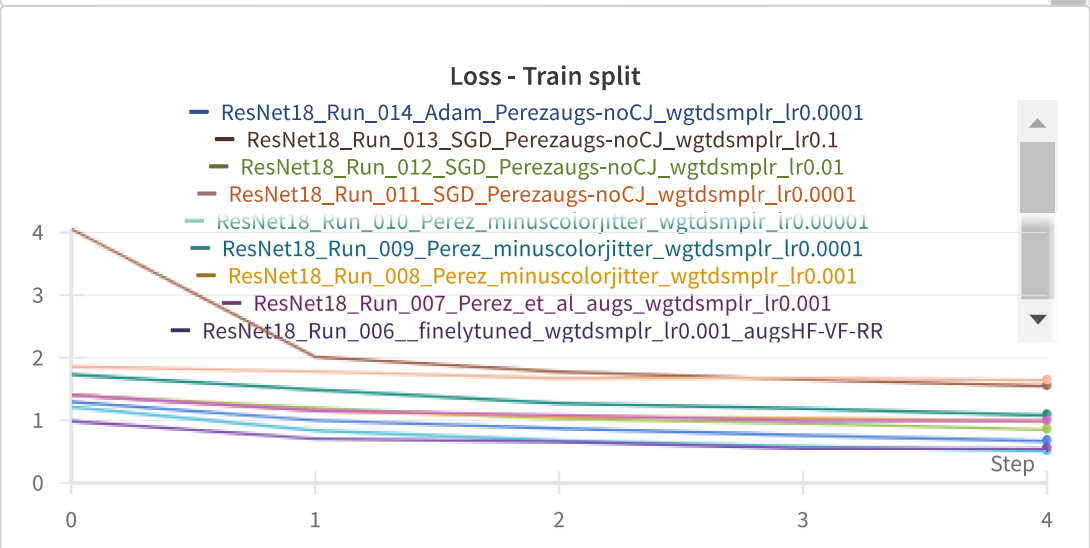
performance on the validation set.

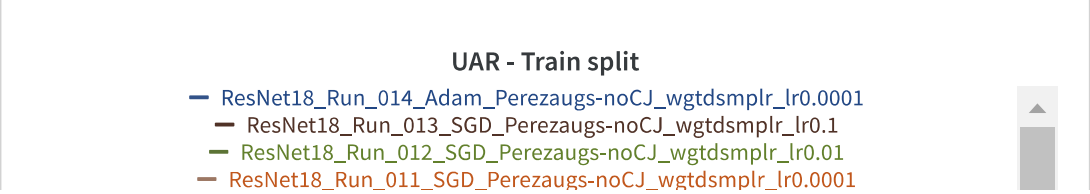
In Run011, Run012 and Run013; a different optimiser was trialled (SGD) with various learning rates (0.0001, 0.01 and 0.1). While the SGD optimiser produced reasonable results when coupled with a learning rate of 0.01, none of the results were as favourable as those obtained by using the Adam optimiser and learning rate of 0.0001 in Run009.

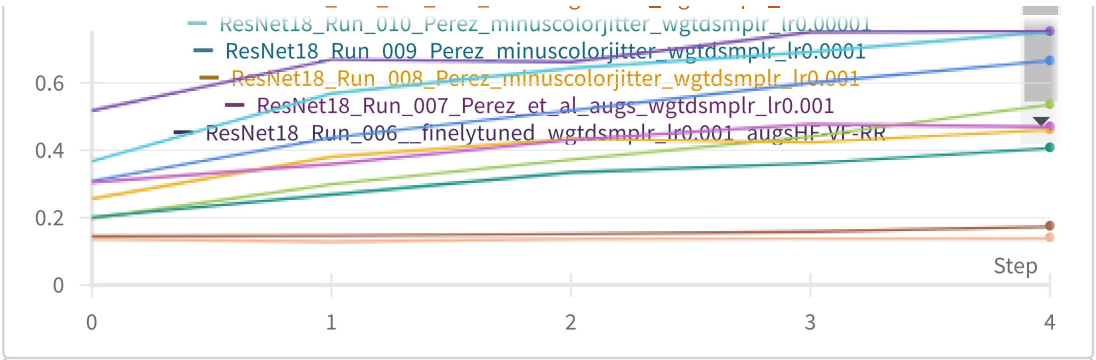
In Run014, Run009 parameters were used again although the probability that the augmentations are applied to the training data was increased from 0.5 to 0.75. Although this resulted in relatively strong performance (Acc = 0.78; UAR = 0.5632; Loss = 0.5506) on the validation set , it did not outperform the previous Run009 performance (Acc = 0.82; UAR = 0.6307; Loss = 0.543)

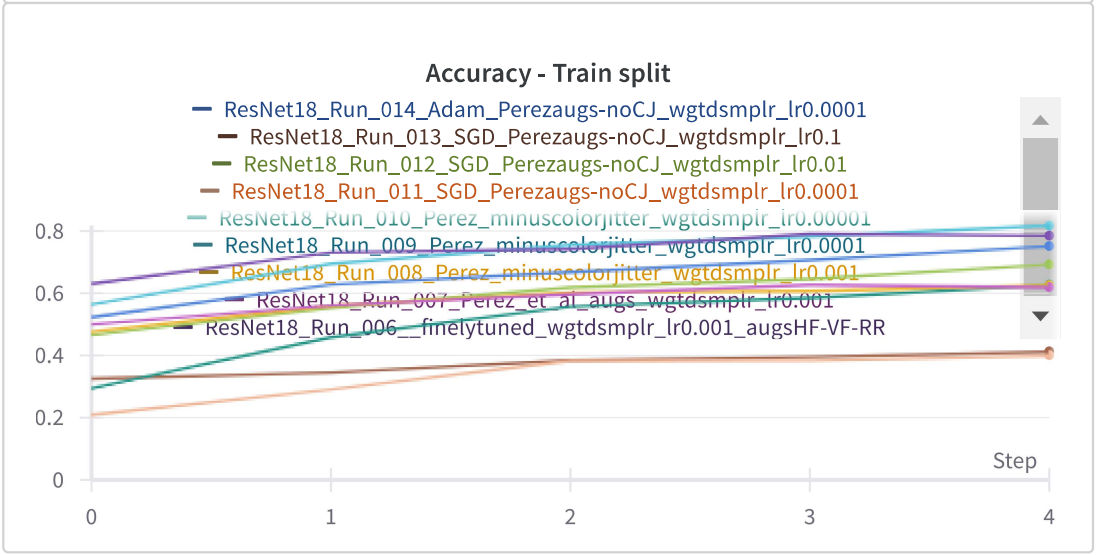
The parameters from Run009 were persisted with due to its superior performance in all of Accuracy, UAR and Loss.

Accuracy, UAR and Loss from the Training split are also included below in graphs for reference.

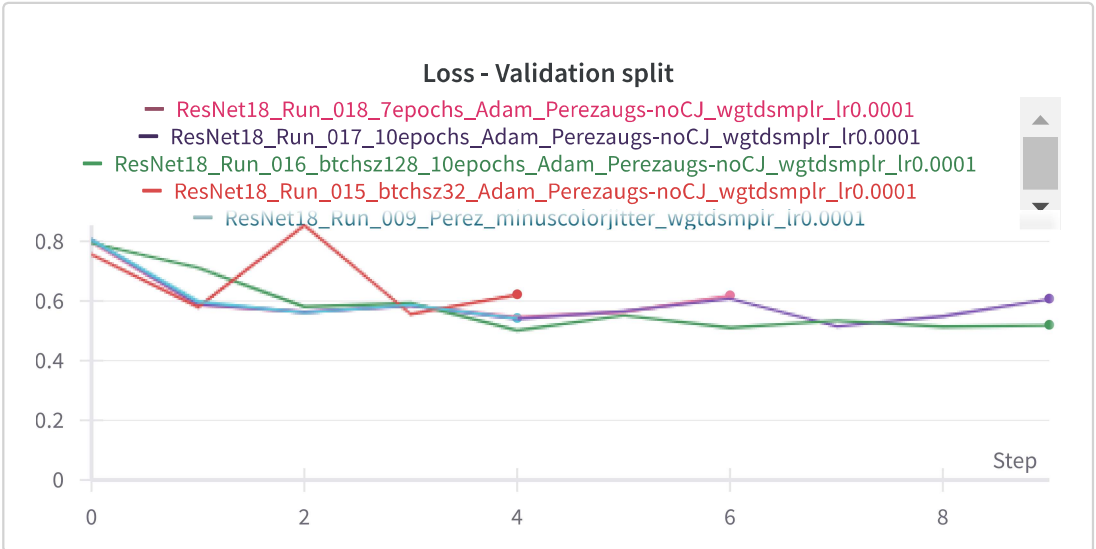


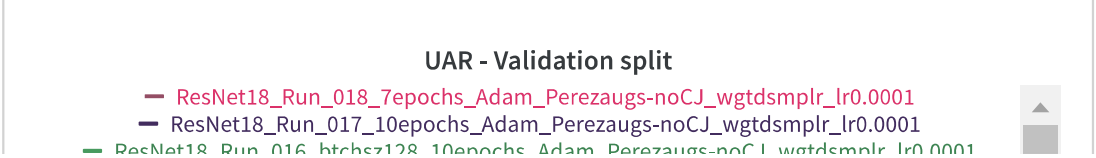


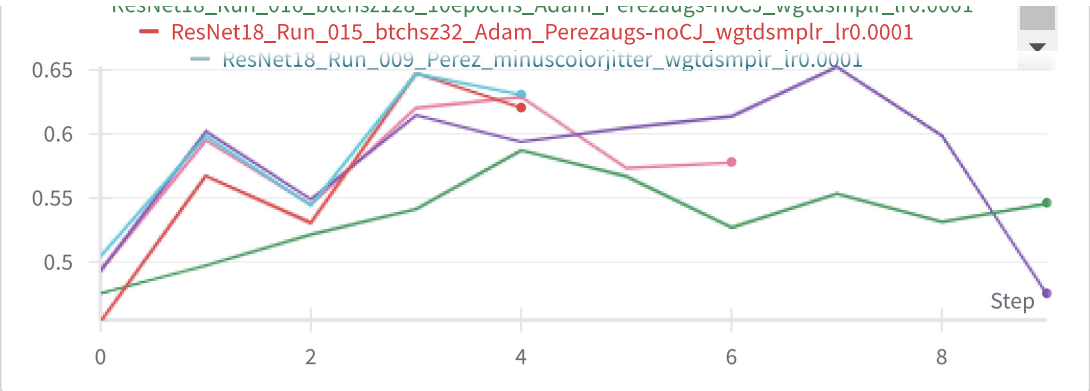


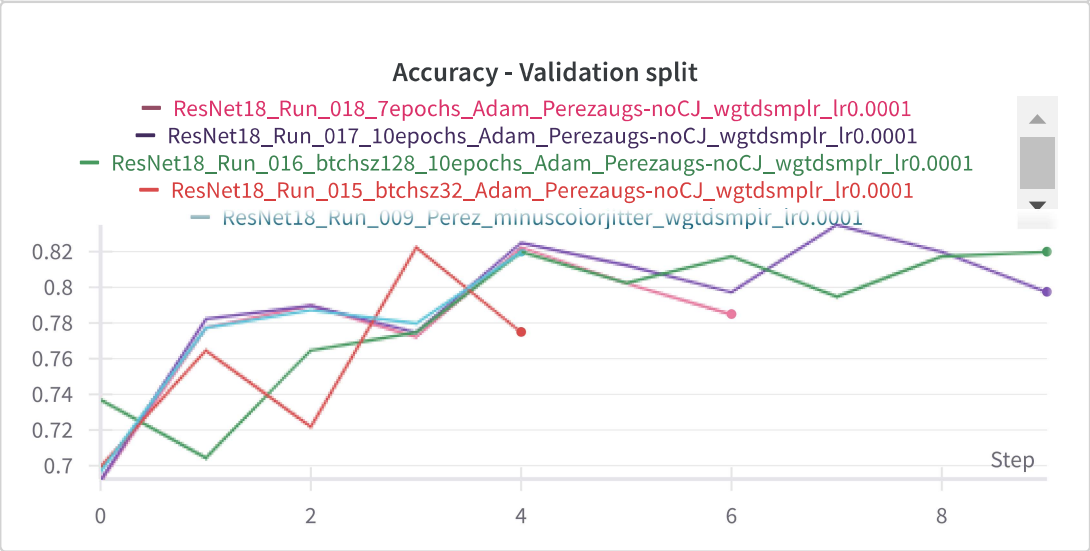


PART 3: Experimenting with epochs and batch sizes









Run009 remained the benchmark (Acc = 0.82; UAR = 0.6307; Loss = 0.543)

Different batch sizes were experimented with. It was expected that halving batch size would double the number of weights updates per epoch and this could improve UAR and ACC.

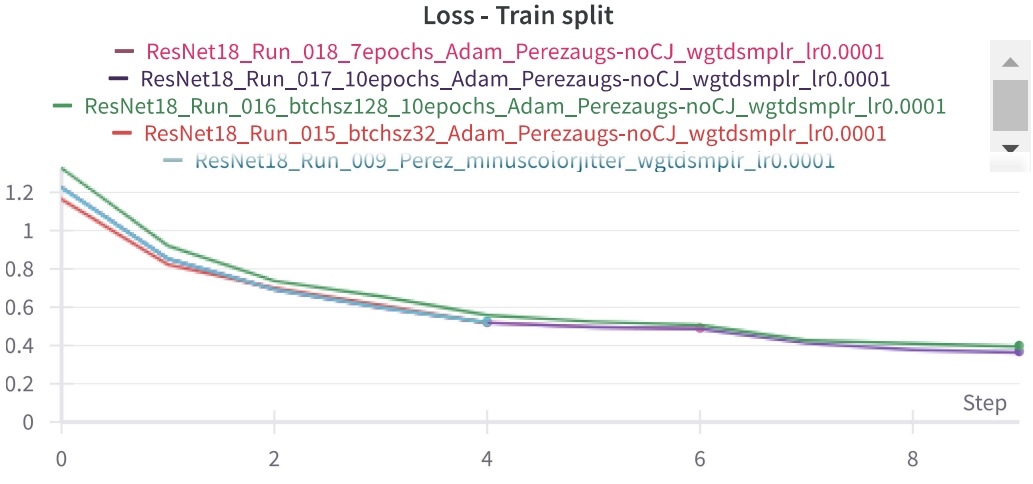
In Run015, when batch size was halved from 64 to 32, accuracy dropped slightly to 0.775 while UAR also decreased slightly to 0.6206. It can be noted on the graphs that a more erratic loss rate and accuracy scores from epoch-to-epoch occurred as a result of this change. This may have resulted from the model updating weights too quickly without considering different features from different representations of the same class. With that said, the model still performed very strongly.

In Run016, a doubled batch size from 64 to 128 was trialled. In this run, the number of epochs was also doubled to 10 so that the model had the same number of opportunities to learn

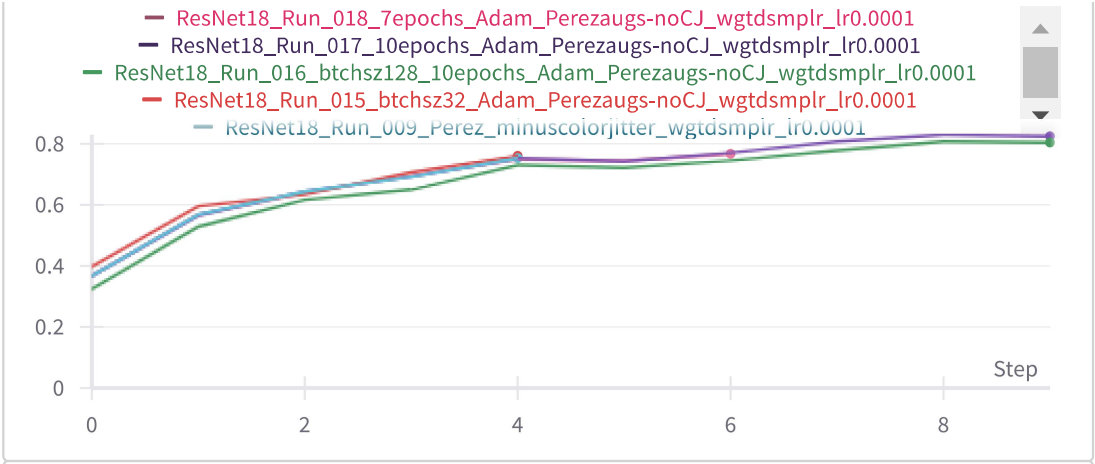
the model had the same number of opportunities to learn different features, despite having less weight updates/batches per epoch. This resulted in the same Accuracy as Run009 (0.82) but a lower UAR (0.5463) on the validation set.

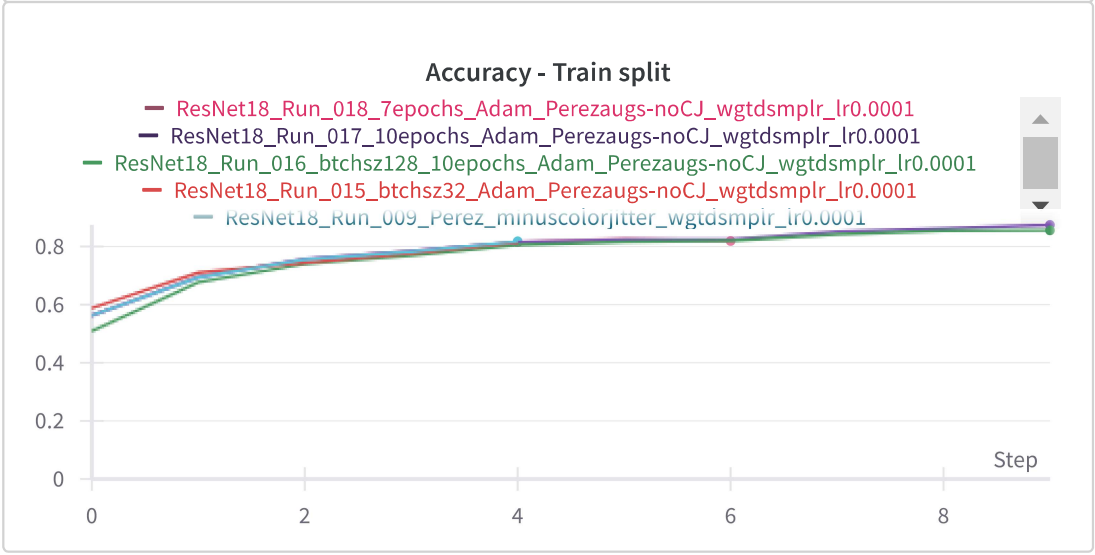
In Run017 and Run018, the batch size was reverted to 64 but more epochs were trialled (10 epochs and 7 epochs). As the model's performance in Accuracy and UAR on the 8th epoch in Run017 but significantly dropped afterwards, 8 epochs should have been chosen in Run018. This was an error. The final results on Run018 were underwhelming (Acc = 0.785; UAR = 0.5781) compared to the original Run009 results using only 5 epochs (Acc = 0.82; UAR = 0.6307) on the validation set. This may not have been the case if 8 epochs were selected rather than 7 as the Run017 trial resulted in a superior Accuracy of 0.835, a superior UAR of 0.6524 and a superior Loss of 0.5167 at the same stage on the validation set.

Nevertheless, in light of the final results on Run018, the model parameters from Run009 were eventually persisted with for the Test split. I was still curious about the results from the halved batch size (Run015), thus this was trialled on the Test split as well, despite the idea that the Test split should only be used once.

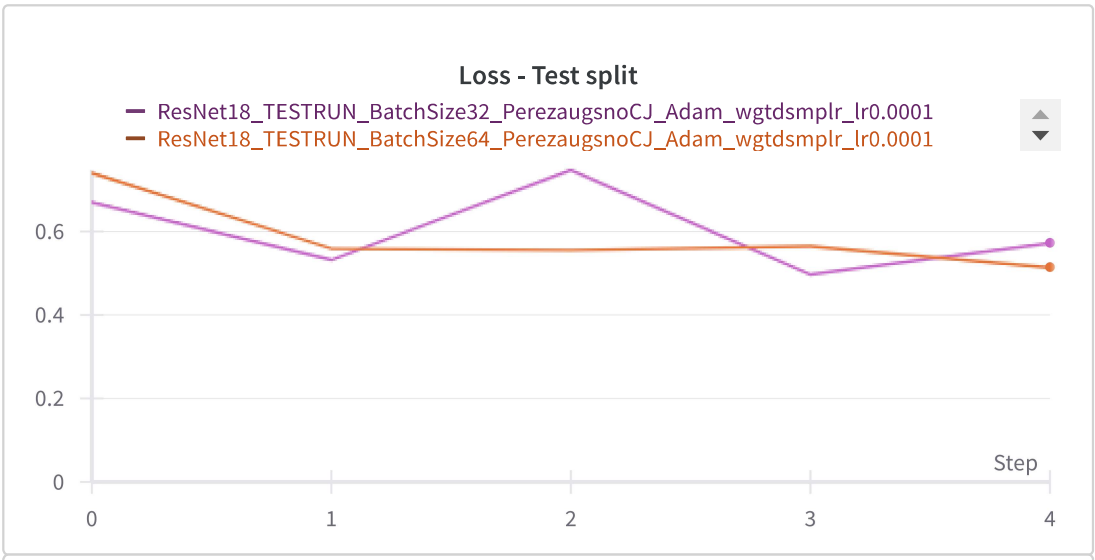


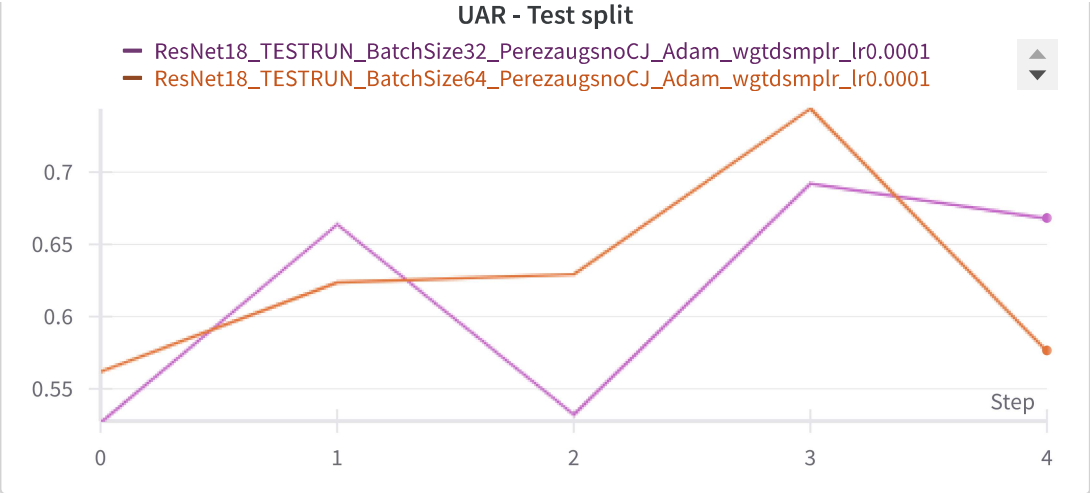
UAR - Train split

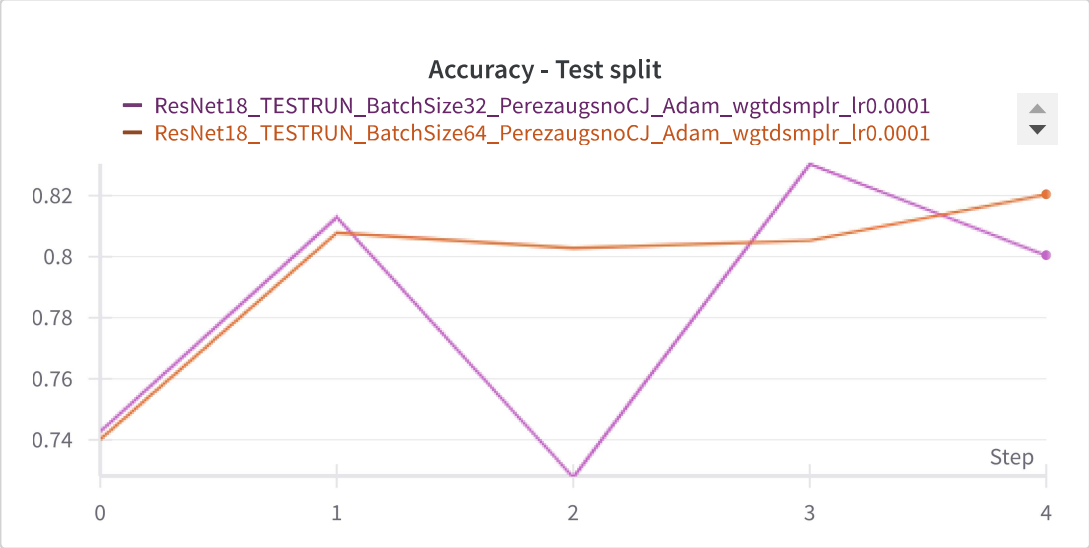




PART 4: Verifying results on the Test split



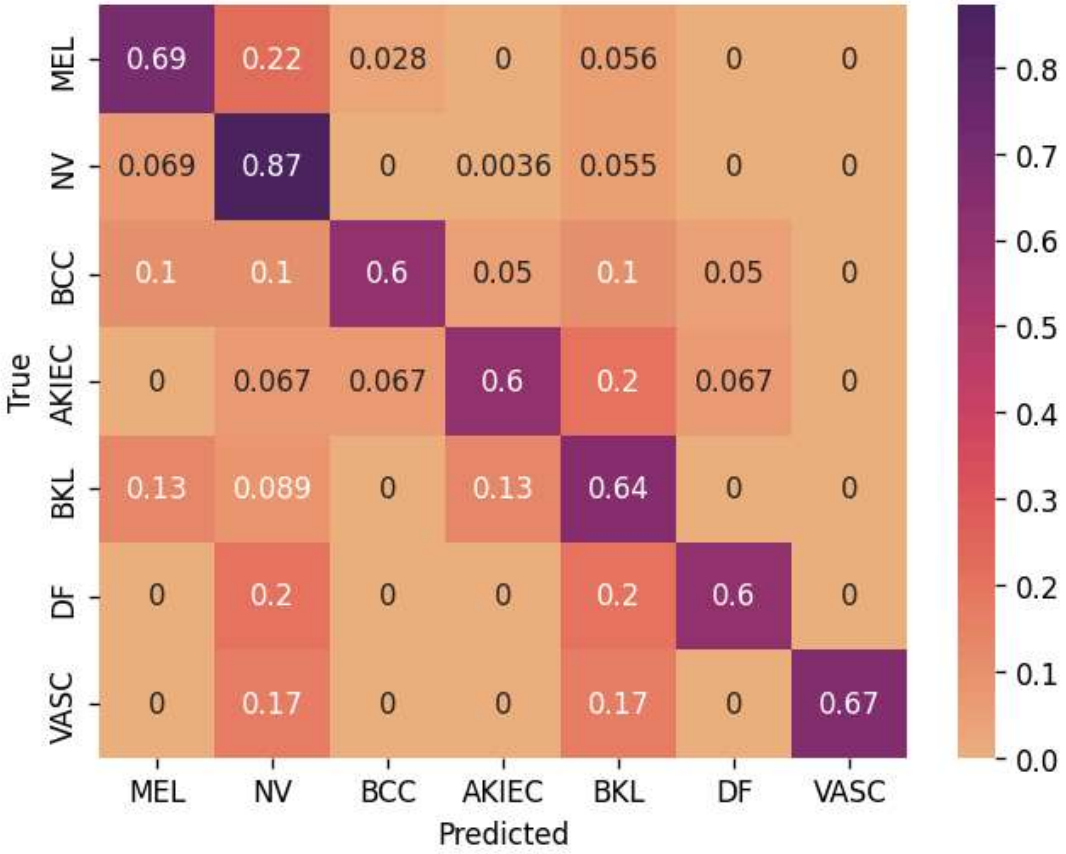




Again, the batch size of 32 resulted in more erratic behaviour as can be seen from epoch-to-epoch on all three plots that graph Accuracy, UAR and Loss.

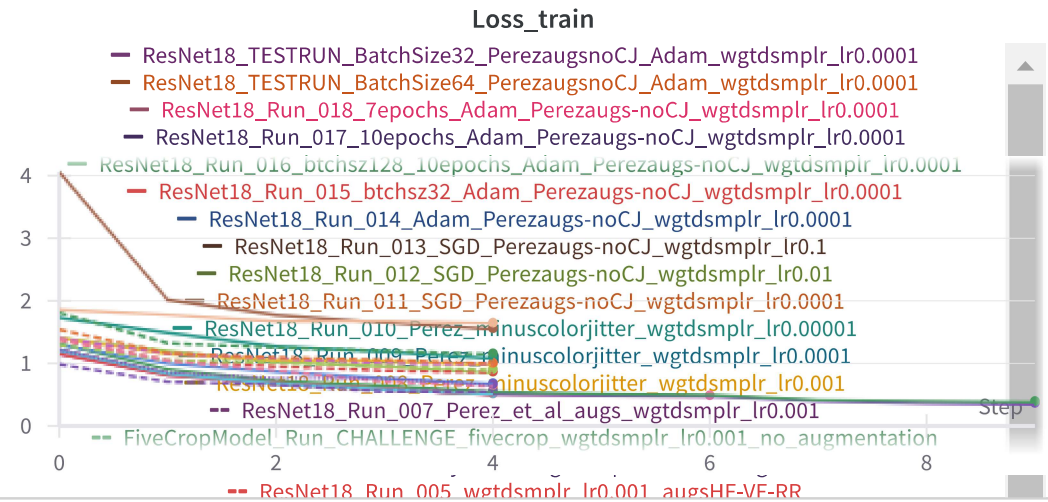
Overall, both the final model using a batch size of 64 (Acc = 0.8204; UAR = 0.5766; Loss = 0.5148) and the final model using a batch size of 32 (Acc = 0.8005; UAR = 0.5766; Loss = 0.5148) performed well on the Test split. It should be noted that perhaps the run could have been stopped after 4 epochs, at both models seemed to peak at this point.

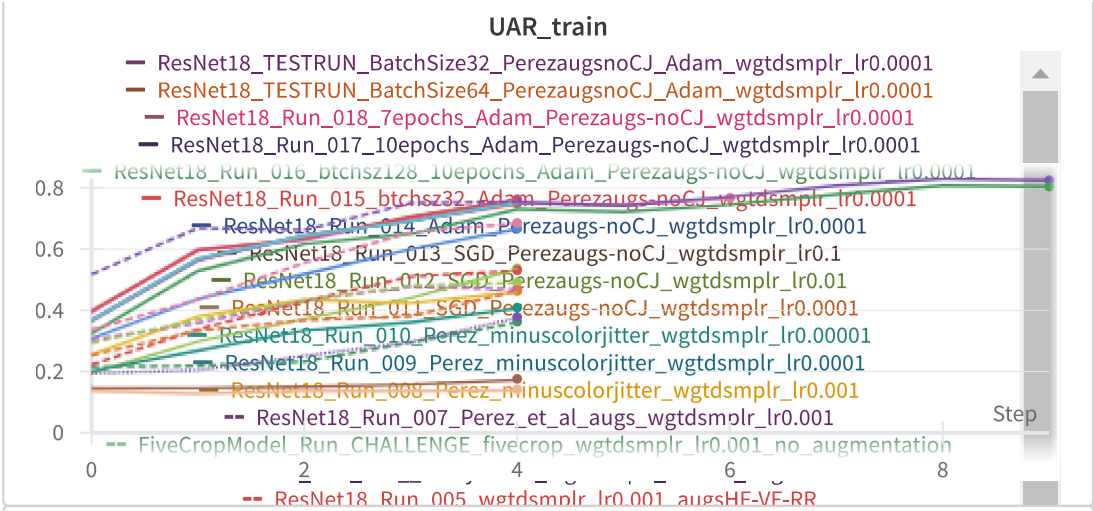
The Confusion Matrix for the model, using a batch size of 32, can be found below.

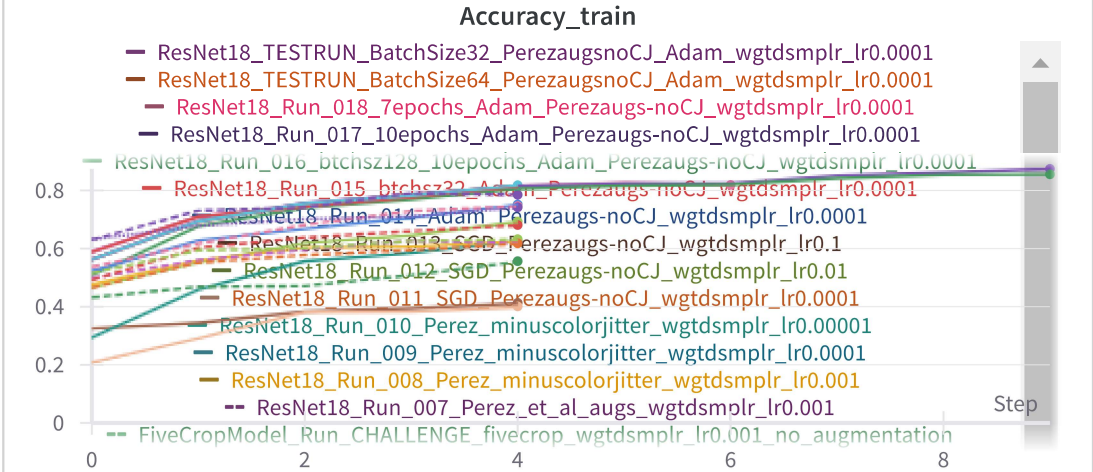


Appendix: Graphs for all 21 runs (training and validation)

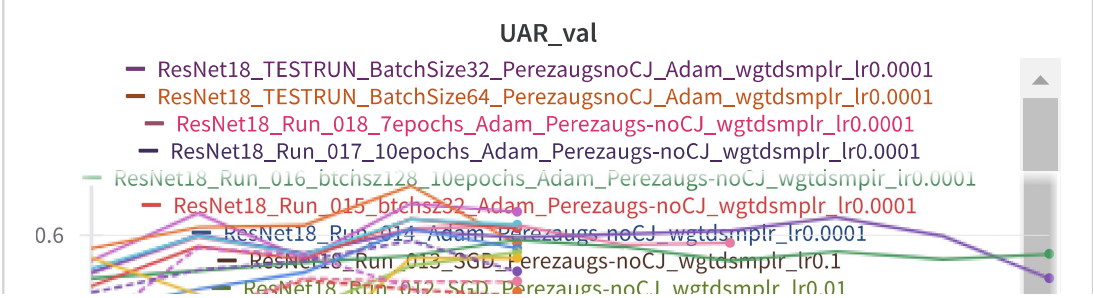
Graphs displaying all 21 runs for both training and validation sets can be found below.




UAR_train
ResNet18_TESTRUN_BatchSize32_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_TESTRUN_BatchSize64_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_Run_018_7epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_017_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_016_btchsz128_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_015_btchsz32_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_014_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_013_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.1
ResNet18_Run_012_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.01
ResNet18_Run_011_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_010_Perez_minuscolorjitter_wgtdsmplr_lr0.00001
ResNet18_Run_009_Perez_minuscolorjitter_wgtdsmplr_lr0.0001
ResNet18_Run_008_Perez_minuscolorjitter_wgtdsmplr_lr0.001
ResNet18_Run_007_Perez_et_al_augs_wgtdsmplr_lr0.001
FiveCropModel_Run_CHALLENGE_fivecrop_wgtdsmplr_lr0.001_no_augmentation
ResNet18_Run_005_wgtdsmplr_lr0.001_augsHF-VF-RR
0.8
0.6
0.4
0.2
0
0
2
4
6
8
Step


Accuracy_train
ResNet18_TESTRUN_BatchSize32_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_TESTRUN_BatchSize64_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_Run_018_7epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_017_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_016_btchsz128_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_015_btchsz32_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_014_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_013_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.1
ResNet18_Run_012_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.01
ResNet18_Run_011_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_010_Perez_minuscolorjitter_wgtdsmplr_lr0.00001
ResNet18_Run_009_Perez_minuscolorjitter_wgtdsmplr_lr0.0001
ResNet18_Run_008_Perez_minuscolorjitter_wgtdsmplr_lr0.001
ResNet18_Run_007_Perez_et_al_augs_wgtdsmplr_lr0.001
FiveCropModel_Run_CHALLENGE_fivecrop_wgtdsmplr_lr0.001_no_augmentation
ResNet18_Run_005_wgtdsmplr_lr0.001_augsHF-VF-RR
0.8
0.6
0.4
0.2
0
0
2
4
6
8
Step


SimpleBNConv_Run_004_wgtdsmplr_lr0.001_augsHF-VF-RR-CC
SimpleBNConv_Run_003_weighted_sampler_lr_0.001_HFVFRRaugs
SimpleBNConv_Run_002_weighted_sampler_lr_0.001_no_aug
Loss_val
SimpleBNConv_Run_001_class_imbalance_lr_0.001_no_aug
ResNet18_TESTRUN_BatchSize32_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_TESTRUN_BatchSize64_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_Run_018_7epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_017_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_016_btchsz128_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_015_btchsz32_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_014_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_013_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.1
ResNet18_Run_012_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.01
ResNet18_Run_011_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_010_Perez_minuscolorjitter_wgtdsmplr_lr0.00001
ResNet18_Run_009_Perez_minuscolorjitter_wgtdsmplr_lr0.0001
ResNet18_Run_008_Perez_minuscolorjitter_wgtdsmplr_lr0.001
ResNet18_Run_007_Perez_et_al_augs_wgtdsmplr_lr0.001
FiveCropModel_Run_CHALLENGE_fivecrop_wgtdsmplr_lr0.001_no_augmentation
ResNet18_Run_005_wgtdsmplr_lr0.001_augsHF-VF-RR
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
Step


UAR_val
ResNet18_TESTRUN_BatchSize32_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_TESTRUN_BatchSize64_PerezaugsnoCJ_Adam_wgtdsmplr_lr0.0001
ResNet18_Run_018_7epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_017_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_016_btchsz128_10epochs_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_015_btchsz32_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_014_Adam_Perezaugs-noCJ_wgtdsmplr_lr0.0001
ResNet18_Run_013_SGD_Perezaugs-noCJ_wgtdsmplr_lr0.1
0.6

References

Perez, F., Vasconcelos, C., Avila, S., & Valle, E. (2018). Data Augmentation for Skin Lesion Analysis. Dans Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis (pp. 303–311). Springer International Publishing. doi:https://arxiv.org/pdf/1809.01442

Created with ❤️ on Weights & Biases.

https://wandb.ai/mfinster/CSE5DL%20Assignment%20Task%201/reports/Skin-lesion-classification-task--Vmlldzo4MTY1OTU2