I began my attempts by experimenting locally with different types of augmentation, optimizers and learning rate schedulers. On my own pc I made a 90/10 train/validation split to estimate results. I also used a separate script to look at the images in the pkl file, so I could see what the dataset actually looked like (for example, I wanted to see if HorizontalFlip would make sense for the contents of the images or not)

1. 67.38% acc, 27m42s <https://www.kaggle.com/code/adrianavlad2/atnn-2025-competition-2-baseline?scriptVersionId=270788714>

The first version uploaded to Kaggle utilized the best options for these operations I was able to find this way.

* For normalization, I used mean = (0.4377, 0.4438, 0.4728), std = (0.1980, 0.2010, 0.1970)
* The optimizer I chose was optim.AdamW(model.parameters(), lr= 0.0005, weight\_decay=1e-3)
* For scheduling I used the first epoch\_nr/10 epochs with linear lr, for warmup, then CosineAnnealingLr.
* I used TTA with horizontal flip and +-20 rotation, with a total of 5 versions created per original image.
* I used EMA for inference.
* I used CrossEntropyLoss with label\_smoothing=0.1
* Batch size = 256, with num\_workers=2 in the DataLoader.
* Epoch\_nr = 50, as RandomErasing was quite slow.
* The augments I found worked best were

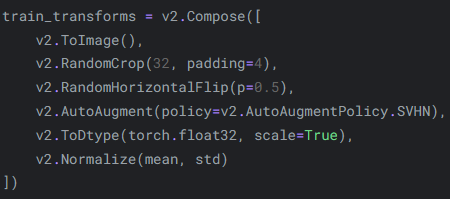
A computer screen shot of a program

AI-generated content may be incorrect.

1. 70.09%, 30m48s <https://www.kaggle.com/code/adrianavlad2/atnn-2025-competition-2-baseline?scriptVersionId=270820004>

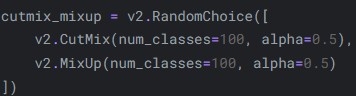
This time I decided to try AutoAugment. Locally I found that v2.AutoAugment(policy=v2.AutoAugmentPolicy.SVHN) worked best, to which I added RandomCrop and HorizontalFlip.

* Epoch\_nr=80 as augments were faster in this version
* Base\_lr changed to 0.001 and weight\_decay to 5e-4, as they seemed to allow for better convergence this time.



1. 74.23% acc, 37m13s <https://www.kaggle.com/code/adrianavlad2/atnn-2025-competition-2-baseline?scriptVersionId=271082755>

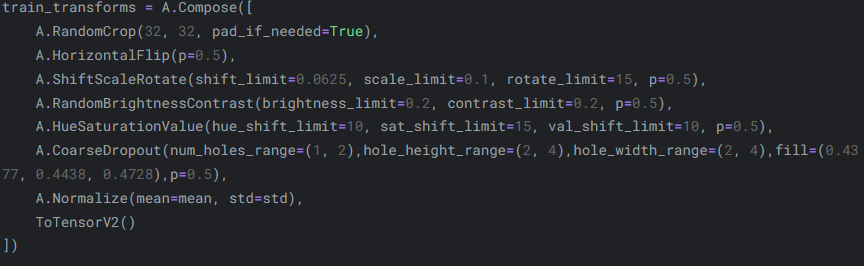
This version I decided to use CutMix and MixUp, with RandomChoice. I tried different alpha values, as well as different probability distributions, including adding a 3rd option to RandomChoice which just returned an unchanged version of the input. In the end the best performing option seemed to be this.



* I set persistent\_workers=True in the DataLoader, though I’m unsure how much it helped.
* Increased label\_smoothing to 0.2 (where previous version suffered from overfitting, this one suffered from underfitting. This change helped)
* Increased epoch\_nr to 100

1. 76.57% acc, 36m48s <https://www.kaggle.com/code/adrianavlad2/atnn-2025-competition-2-baseline?scriptVersionId=271106303>

Decided to try the albumentations library, which gave similar results, but was much faster, allowing for more epochs. Again, I tried multiple operations from this library before reaching this final combination of augments.



* Epoch\_nr increased to 150.
* TTA rotation changed to +-10, though I’m unsure what difference that made as I never tested this change on its own.