Final Poster - Vision

Yigit Deniz Celebi, Alper Ari, Bora Arda Zeytinoglu
abd-vision

Modality 2/2

Motivation

- To build an AutoML pipeline, which aims to train an image classification model with any given dataset.
- To increase model's accuracy on test dataset while fine tuning its hyperparameters using scientific approaches.
- To find balance between training time and top-1 accuracy score.
- To apply multiprocessing and parallelization techniques to speed up exploration in configure space
- To apply data augmentations to create data variations that helps the model improve accuracy score

Planning

- Choosing (suitable) datasets/training splits
- Data augmentation
- Finding the best epochs
- Fine tuning hyperparameters
- Splitting the data 80/20 for training and validation purposes
- Choosing best model to balance speed and accuracy
- Feeding the data into the model for final training
- Evaluating the predictions and obtaining the top-1 accuracy score

- Week 1
- Week 2
- Week 3
- Week 4
- Week 5
- Week 6
- Week 7
- Week 8
- Week 9
- Week 10
- Bonus
- Literature

Coding Approach

Literature & Research

- **Before everything, we tried to find a promising and easy-to-use AutoML library from the internet and decided on Neps.**
- **We did research about how CNNs work, types of layers and the difference between popular networks such as ResNet, VGG, ShuffleNet, MobileNet, EfficientNet, etc.**
- **Depending on the CNN we chose, we also determined the data augmentation techniques before we feed our data into the model.**
- **Before we started implementing HPO speed up techniques like Bayesian Optimization, Successive Halving, Hyperband and Priorband, we went through their scientific papers to understand how they actually work.**
- **We also did a research about learning rate schedulers by going through PyTorch's documentation. We ended up using OneCycleLR and ReduceLROnPlateau schedulers and adding them to our configuration space.**
- **We tried several plotting libraries to visualize and analyze prediction results.**

Implementation

1- We started with implementing AutoML class with its member variables and functions.
2- Then we built the base of our pipeline code inside AutoML class which is to be callable and generic.
3- We came up with several CNNs with their hyperparameters and structured our configuration space.
4- After making sure that our automated pipeline works with Tiny VGG i.e. it successfully evaluates configurations and finds the best one, we decided to use pre-trained models like MobileNetV2 for faster training and better accuracy.
5- We introduced various data augmentation transforms and extended our configuration space.
6- We added multiprocessing feature to our pipeline to explore more configurations in a unit time.
7- First we used hyperband to speed up the training, then switched to priorband by giving higher confidence to epochs.
8- We introduced learning rate scheduler to our pipeline for the sake of dynamic algorithm configuration.

Resources Used

For development:
- 2 GTX1650 GPU
- 1 RTX4070Ti GPU
- 2 Intel Core CPUs
- 1 AMD CPU
- Total compute estimate: 600 CPU-h

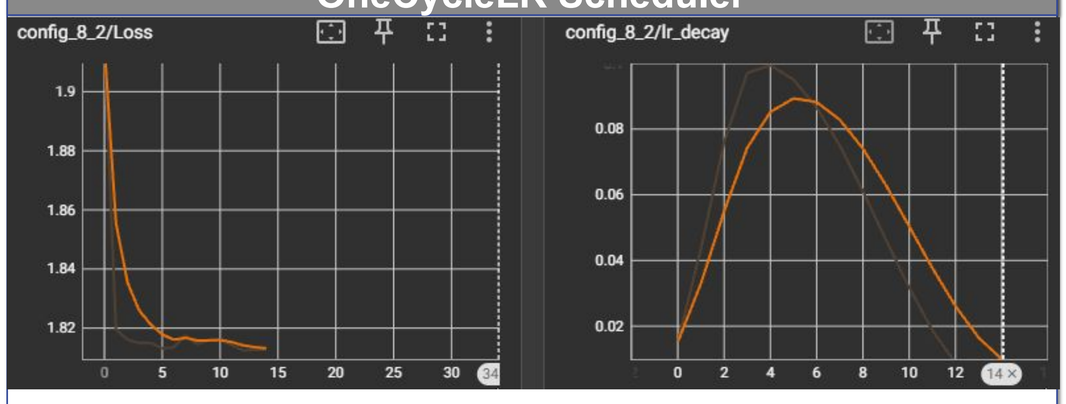
For AutoML:
- Google Colab (A100 + L4)
- 12h

Workforce:
- 1 full week on average

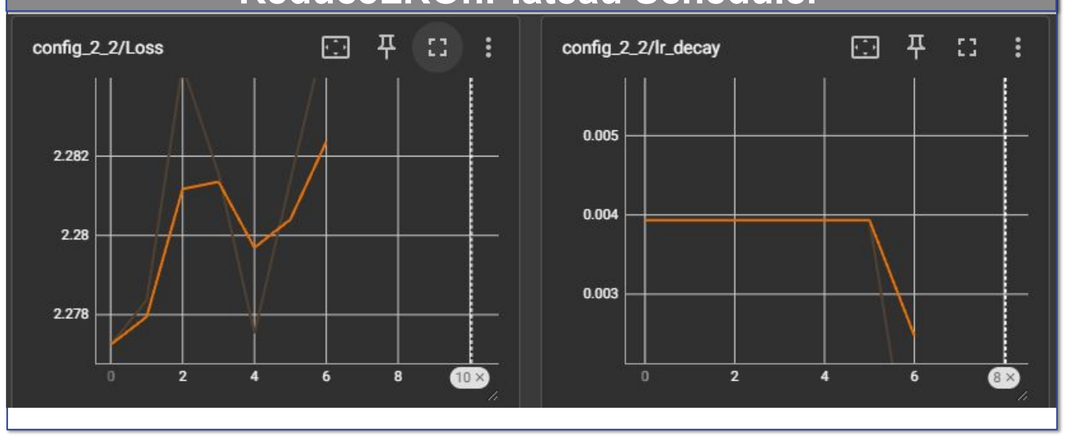
Number of queries for test score generation: **3**

Empirical Results

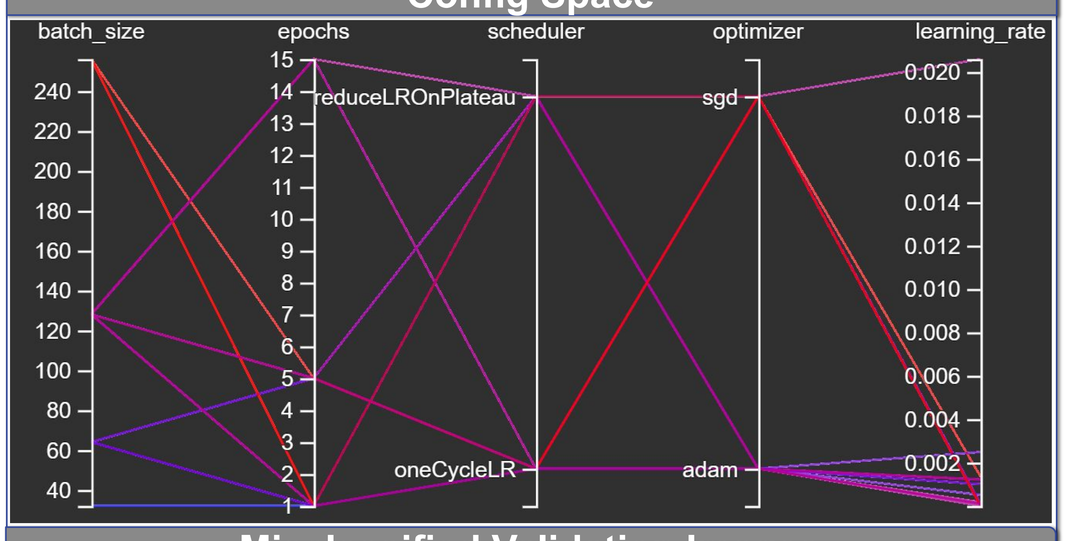
OneCycleLR Scheduler



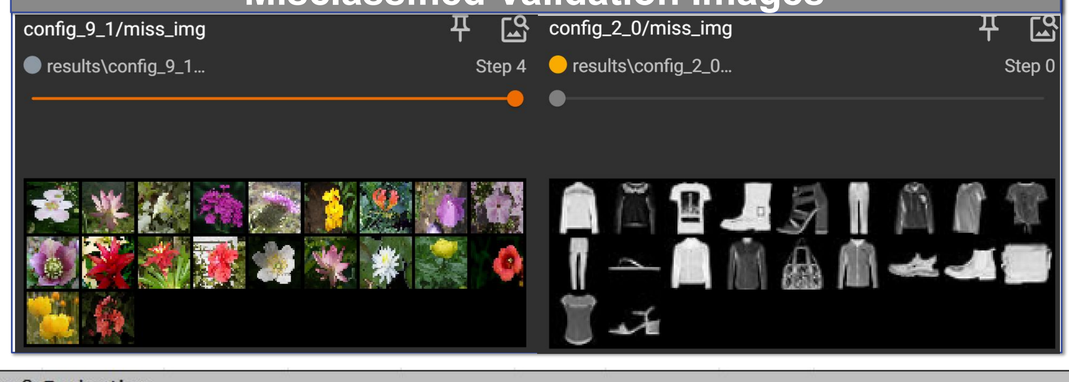
ReduceLROnPlateau Scheduler



Config Space



Misclassified Validation Images



Best configuration & Evaluation

	epochs	batch size	optimizer	scheduler	time elapsed (s)	LR	last LR	loss	top_1_acc	top_5_acc	precision	recall	F1	device	total pipeline runtime (s)
Emotions	15	64	ADAM	oneCycleLR	1007	0.02094	0.0099	1,8126	0.24	0.86	0.035	0.14	0.05	A100	5578
Flowers	5	128	ADAM	oneCycleLR	201	0.0773	0.0273	3,8777	0.9678	0.9963	0.9660	0.961	0.962	A100	3150
Fashion	1	64	ADAM	reduceLROnPlateau	258	0.0039	0.0039	2,2772	0.7996	0.993	0.8039	0.7996	0.7993	L4	15512
Skin Cancer	15	32	SGD	oneCycleLR	616	0.01525	0.0099	1,1424	?	?	?	?	?	A100	3825