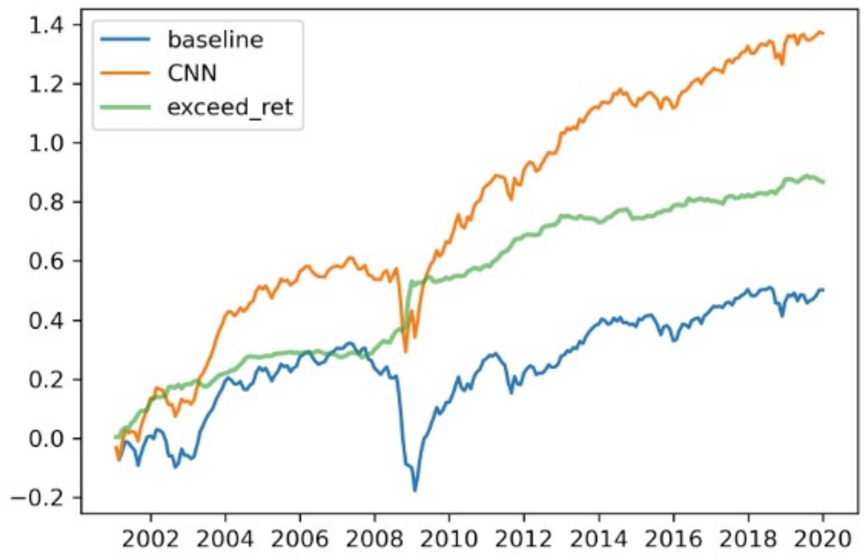
Implementation Overview

- Guided implementation by github user lich99 → https://github.com/lich99/Stock_CNN
- 20-day stock charts provided by authors (5-day and 60-day data omitted)
- Goal to train model and compare output probability logits to the cited paper
  - Focusing on I20/R5 and I20/I20 and comparing results with both equal weight and value weighted High-Low Portfolios

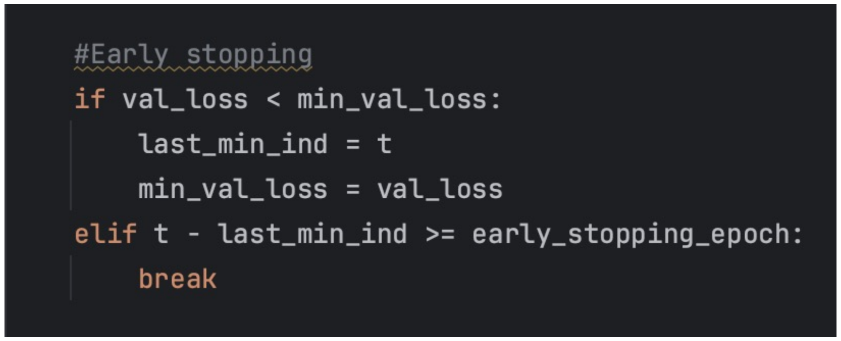
Github results



Train & Test

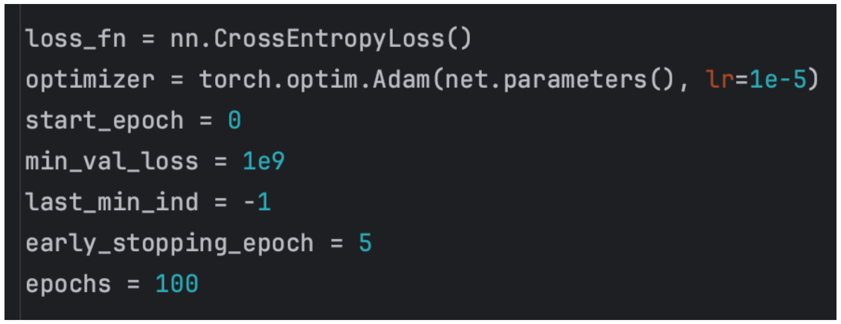
- Sean's implementation trained on **all** data
  - We are somewhat constrained by computational power → each Epoch w/ GPU took ~45 minutes
  - Years used for training == range(1991, 2001) == 273,000 images
  - Full training time == 3.2 hours before early stopping kicked in at 6th epoch
  - Final loss → **testing=** .685 **val=** .745

Early stopping



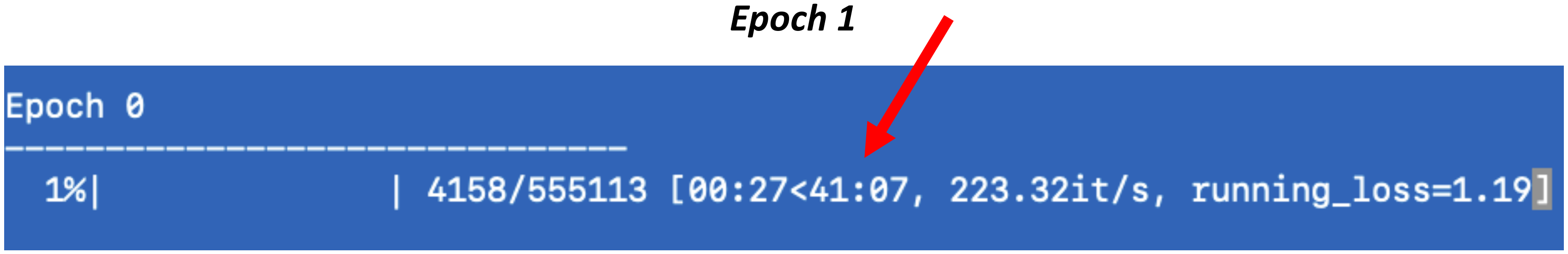
```
#Early stopping
if val_loss < min_val_loss:
    last_min_ind = t
    min_val_loss = val_loss
elif t - last_min_ind >= early_stopping_epoch:
    break
```

Parameters



```
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(net.parameters(), lr=1e-5)
start_epoch = 0
min_val_loss = 1e9
last_min_ind = -1
early_stopping_epoch = 5
epochs = 100
```

Epoch 1

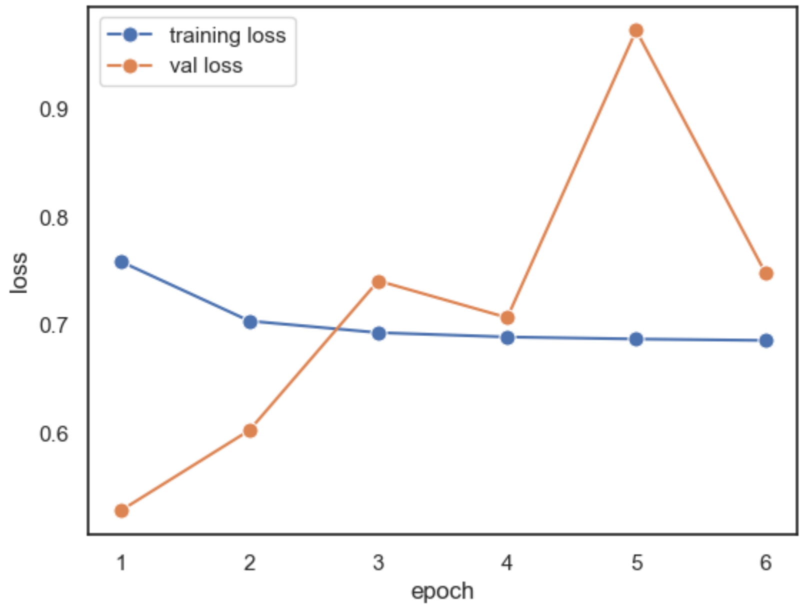


```
Epoch 0
-------------------------------
  1%|          | 4158/555113 [00:27<41:07, 223.32it/s, running_loss=1.19]
```

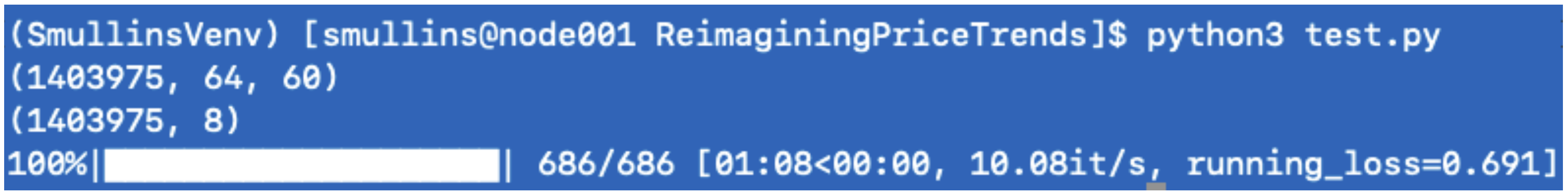
Train & Test (Cont…)

- Error
  - Train/val error is confusing. Model stops after two consecutive higher val error. Something does not seem right…
- Test
  - Implementation tested on **all** data (years 2001-2019)
  - Testing and logit predictions had runtime of 59 seconds with a final test error of .69
  - 1.4MM images tested

Train & Val Error



Testing

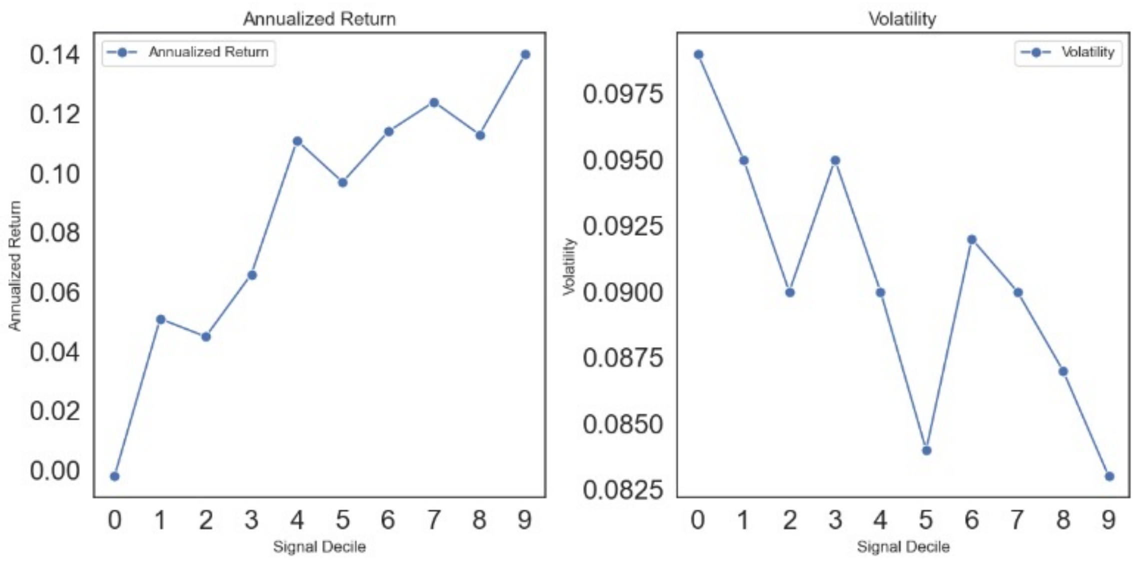


```
(SmullinsVenv) [smullins@node001 ReimaginingPriceTrends]$ python3 test.py
(1403975, 64, 60)
(1403975, 8)
100%|██████████████████████| 686/686 [01:08<00:00, 10.08it/s, running_loss=0.691]
```

Evaluation & Comparison

- Positive linear relationship between return and decile
  - 14% return on top (9)
  - ~0% return on bottom (0)
  - In-between deciles monotonically increasing
- Volatility in line with paper → it is not massively influenced by decile
- Smullins implementation missing **potency** in identification of returns
  - Model able to discern returns with lowest decile 0% RET and highest decile 14%, but paper comparison we are missing potency of tail ends (eg: -32% return for decile 1)

Smullins Equal Weighted Return & Vol



Paper vs. Smullins I20/R5

	Paper		Smullins Implementation		Difference
Decile	**Ret**	**SR**	**Ret**	**SR**	**Ret**
1	-32.0%	-1.94	0.0%	-0.12	32.0%
2	-4.0%	-0.21	5.1%	0.43	9.1%
3	4.0%	0.20	4.5%	0.39	0.5%
4	8.0%	0.43	6.6%	0.59	1.4%
5	12.0%	0.65	11.1%	1.12	0.9%
6	15.0%	0.80	9.7%	1.04	5.3%
7	19.0%	0.97	11.4%	1.13	7.6%
8	23.0%	1.19	12.4%	1.27	10.6%
9	27.0%	1.40	11.3%	1.18	15.7%
10	52.0%	2.76	14.0%	1.57	38.0%

Thoughts

- Very impressed with the model → able to discern deciles monotonically increasing and sort (14% versus 0%)
- I am confused between value weight and equal weight, our results look almost exact to their **value weighted,** High-Low (10-1) portfolio
- I feel something is wrong with the train/val process
    - Is training supposed to be that long w/ GPU utilization (40min per epoch)?
    - Val error does not make much sense as it jumps around, BUT the last epoch performs best on the test set
    - Shouldn't we have more epochs for training? Parameter set to 100 but early stopping at 6…
- Paper vs. Smullins difference could be the way I am splitting the logit outputs → We are using 10 equal-frequency bins in line with the paper? What if we used bins by weight or return? This may help potency