Human Activity Recognition (HAR) Transfer Learning

Mica Haney
Kushal Patel
Vrushabh Ajaybhai Desai
Mayur Vora
Rhea Chitturu
Sreeja Bellamkonda

Team Members

Name	Email	Role
Mica Haney	micahaney@my.unt.edu	Team Leader
Kushal Patel	kushalpatel2@my.unt.edu	Team Member
Vrushabh Ajaybhai Desai	VrushabhAjaybhaiDesai@my.unt.edu	Team Member
Mayur Vora	Mayuriswarbhaivora@my.unt.edu	Team Member
Rhea Chitturu	rheachitturu@my.unt.edu	Team Member
Sreeja Bellamkonda	Sreeja.Bellamkonda@my.unt.edu	Team Member

Abstract

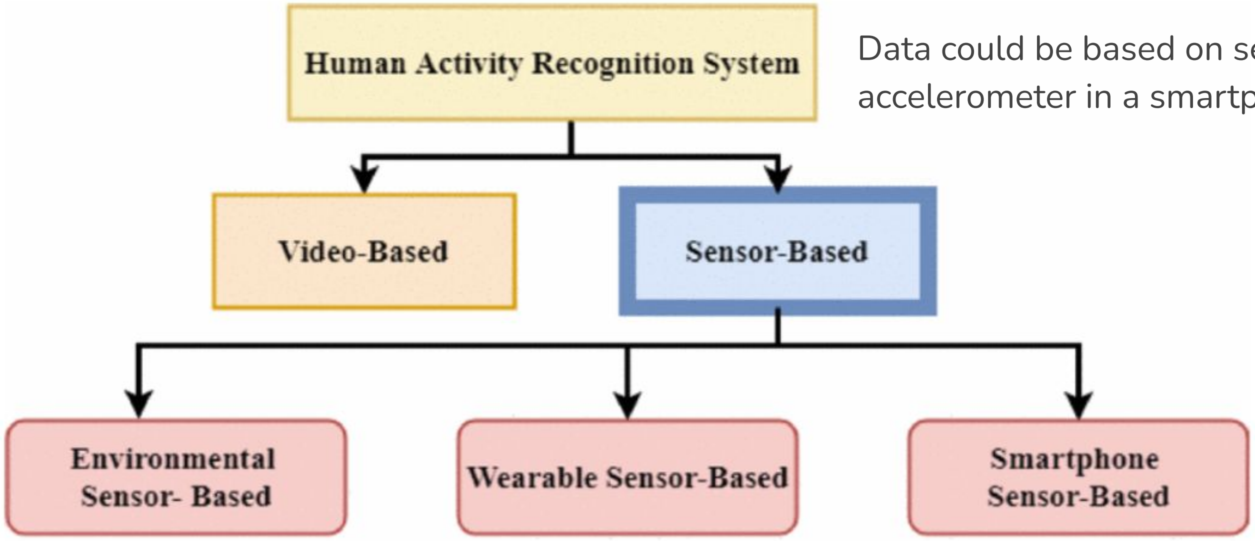
In the previous iteration of this project, using transfer learning in the domain of human activity recognition (HAR) was performed. The project specifically looked at adding in new activities, using data from the same dataset. This allows for the model to look only at the activities without having to decide if the activity comes from one collection method or another.

This iteration of the project proposes to tackle that problem, and see if transfer learning is a valid approach to working with multiple datasets. If so, this would indicate a way for handling real-world problems where data collection methods might differ between locations and patients due to obstacles such as sensor availability, safety restrictions, collection protocol, etc.

Introduction: HAR

Human Activity Recognition (HAR) is predicting the movement of a person such as walking, talking, standing, and sitting.

Data could be based on sensor data, such as an accelerometer in a smartphone or wearables or video.



Common Challenges of HAR

Below are aspects of HAR data that often cause significant issue when training models.

- Diversity of age, gender, and number of subjects.
- Postural transitions (Sitting to Standing, Walking to Jogging etc).
- Number of sensors and types of sensors.
- Missing values or labeling errors.
- Complex activities.
- Sparse data.

Introduction: Transfer Learning

Transfer learning is a form of fine-tuning.

Only certain layers are updated, the rest are frozen and the weights don’t update.

This allows for the model to adjust to the new dataset by leveraging the pre-trained model’s knowledge of the original dataset without losing any information contained in the pre-trained weights.

The dataset pre-training is done on is the **source** dataset.

The dataset fine-tuning is done on is the **target** dataset.

The target dataset must closely resemble the source dataset.

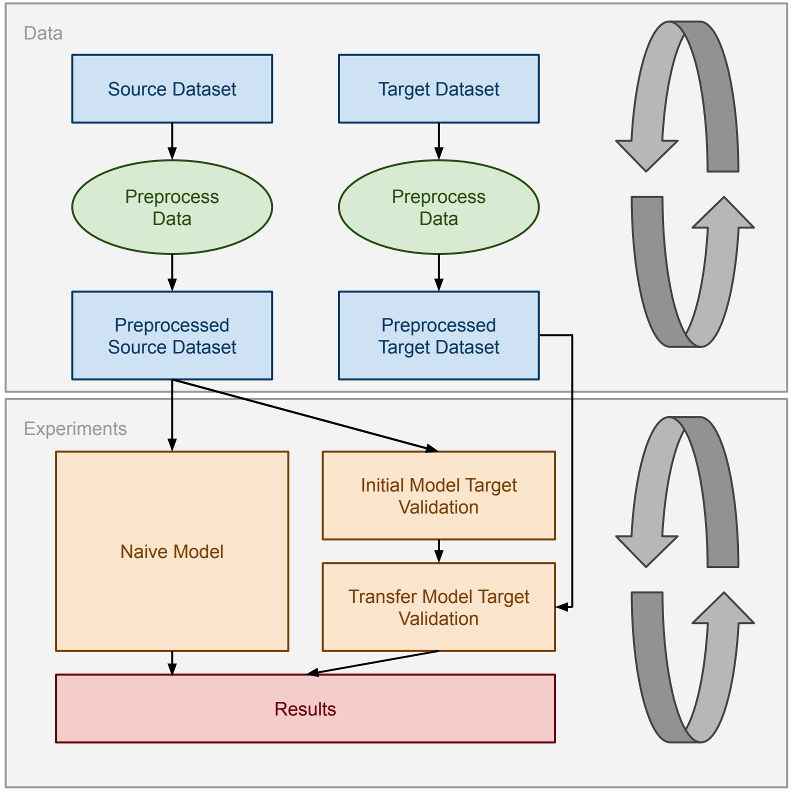
Design

All code is written in Google Colab, with the data stored in Google Drive.

The code is written in python, and largely uses the tensorflow, sklearn, pandas, numpy, and pyplot libraries.

Data will be analyzed then preprocessed only once, initially. This will be repeated if experimentation requires it and time allows.

Experiments will be run in sets of 5, with a naive, two initial/source, and two transfer models.


Data
Source Dataset
Target Dataset
Preprocess Data
Preprocess Data
Preprocessed Source Dataset
Preprocessed Target Dataset
Experiments
Naive Model
Initial Model Target Validation
Transfer Model Target Validation
Results

Datasets

Name	Users	Shape	Activites
Wearable HAR	22	(11,600, 121)	7 (jump, lie down, sit, stand, stairs down, stairs up, and walk)
UCI HAR	30	(20,598, 562)	6 (lie down, sit, stand, stairs down, stairs up, and walk)
UCI HAPT	30	(20,432, 562)	6 (lie down, sit, stand, stairs down, stairs up, and walk) + 2 (stand-to-sit, sit-to-stand)
UniMiB	30	(12,428, 454)	9 (jump, lie down, sit, stand, stairs down, stairs up, and walk, Jogging)
WISDM	36	(4,774, 406)	6 (Walking, Jogging, Upstairs, Downstairs, Sitting)

Data

We chose the UCI HAR dataset to be our source dataset, and the WISDM dataset to be the target. These selections were made purely based on sample counts.

UCI HAR:

1. 34,444 sampes
2. 3,088 samples
3. 2,912 samples
4. 3,554 samples
5. 3,812 samples
6. 3,888 samples

Total: 20,598 samples

WISDM:

1. 456 sampes
2. 1,557 samples
3. 571 samples
4. 1,728 samples
5. 264 samples
6. 198 samples

Total: 4,774 samples

Model

2 model architectures were initially constructed, a dense network and a CNN. However, due to time constraints only the dense network was used.

Dense Architecture:

- Input layer
- 7 fully connected layers, starting at 1024 hidden units and halving the count each successive layer
- Output layer, softmax activation

Experimental Setup

Each architecture was trained three times per experiment.

- One “naive” baseline of end-to-end training.
- One source model for transfer learning.
- One target model via transfer learning.

Each experiment was run five times and the average scores were reported.

Experimental Setup (Cont.)

Thirty-nine experiments were run:

1. Baseline
2. Drop Learning Rate by 1
3. Drop Learning Rate by 2
4. Raise Learning Rate by 1
5. 2 Unfrozen Layers
6. 3 Unfrozen Layers
7. 4 Unfrozen Layers
8. 5 Unfrozen Layers
9. 6 Unfrozen Layers
10. 7 Unfrozen Layers
11. 8 Unfrozen Layers
12. 5 Samples of Each Target Class Kept
13. 10 Samples of Each Target Class Kept
14. 15 Samples of Each Target Class Kept
15. 20 Samples of Each Target Class Kept
16. 25 Samples of Each Target Class Kept
17. 30 Samples of Each Target Class Kept
18. 35 Samples of Each Target Class Kept
19. 40 Samples of Each Target Class Kept
20. 45 Samples of Each Target Class Kept
21. 50 Samples of Each Target Class Kept
22. 55 Samples of Each Target Class Kept
23. 60 Samples of Each Target Class Kept
24. 65 Samples of Each Target Class Kept
25. 70 Samples of Each Target Class Kept
26. 75 Samples of Each Target Class Kept
27. 80 Samples of Each Target Class Kept
28. 85 Samples of Each Target Class Kept
29. 90 Samples of Each Target Class Kept
30. 95 Samples of Each Target Class Kept
31. 100 Samples of Each Target Class Kept
32. 105 Samples of Each Target Class Kept
33. 110 Samples of Each Target Class Kept
34. 115 Samples of Each Target Class Kept
35. 120 Samples of Each Target Class Kept
36. 125 Samples of Each Target Class Kept
37. 130 Samples of Each Target Class Kept
38. 135 Samples of Each Target Class Kept
39. 140 Samples of Each Target Class Kept

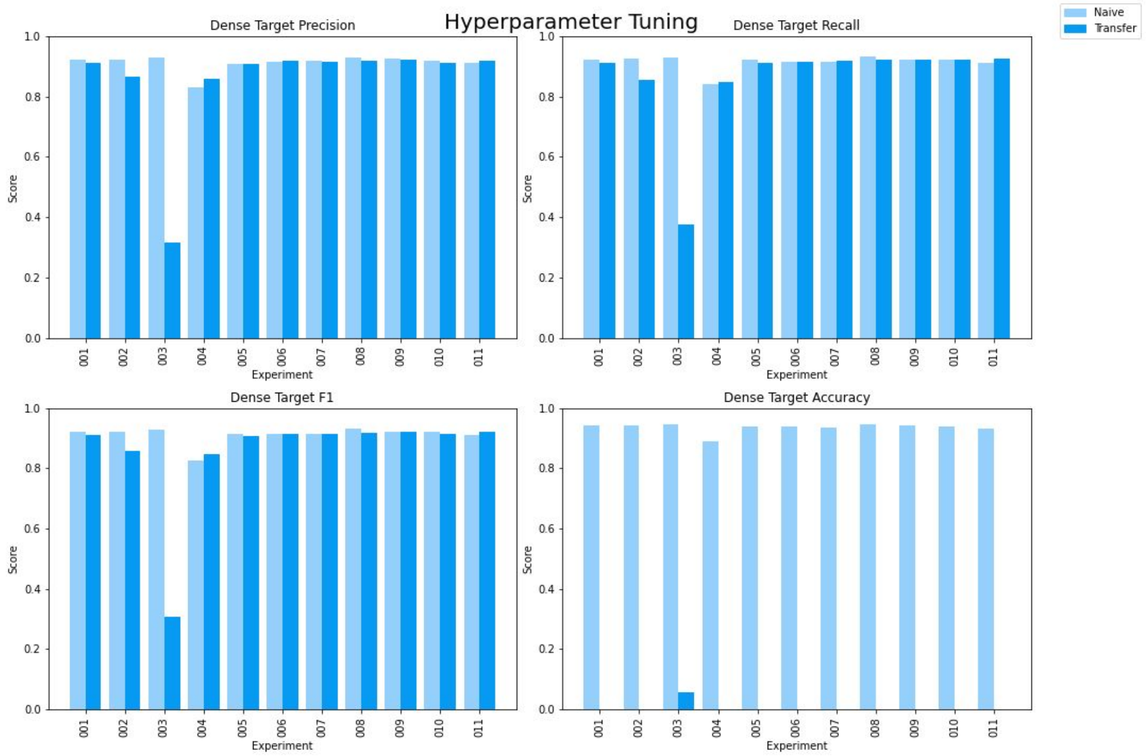
Training

Hyperparameters:

- 40 epochs per naive model, 20 epochs per initial and per transfer model.
- 5 runs per model training.
- Learning rates of $1e^{-3}$ or $1e^{-5}$. Settled on $1e^{-3}$ after hyperparameter tuning.
- Freeze all but 1 to 8 layers for transfer learning. Settled on 6 unfrozen layers after hyperparameter tuning.

Results: Hyperparameter Tuning

The first 11 experiments were done to tune the hyperparameters. Learning rate had a significant impact on transfer learning scores, however the differences between the scores of the number of layers left unfrozen is negligible.

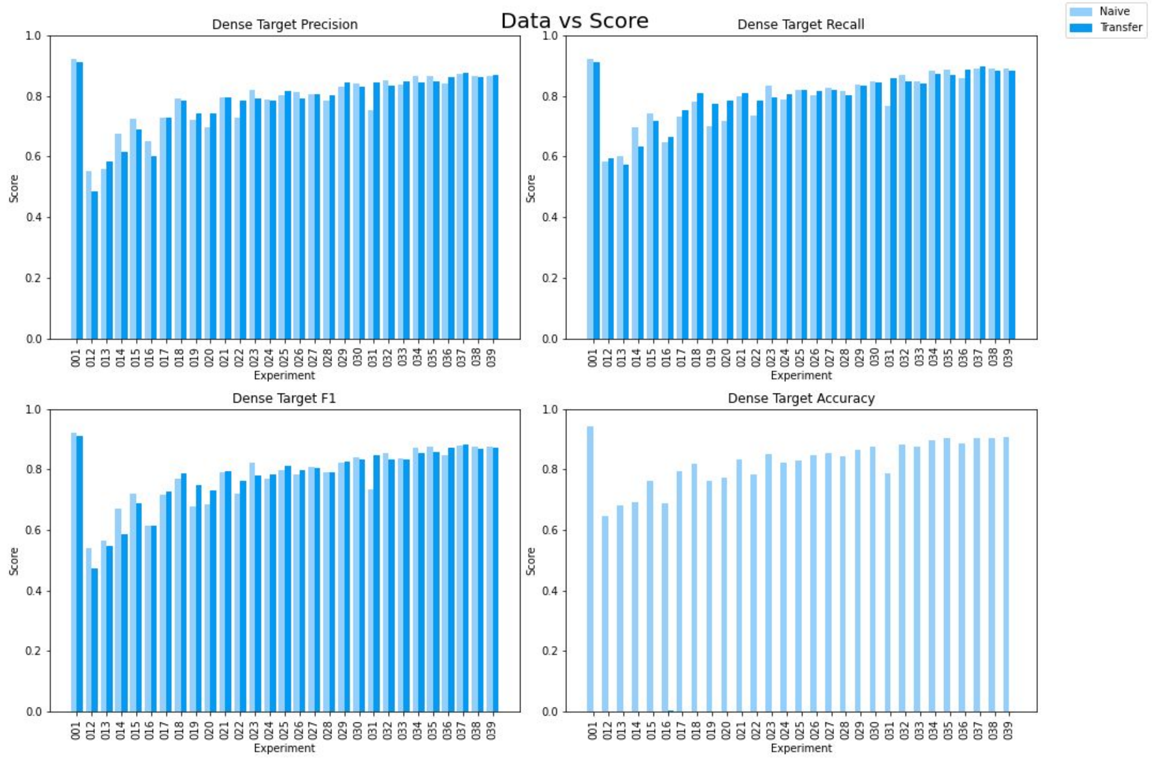


Results: Data in Target

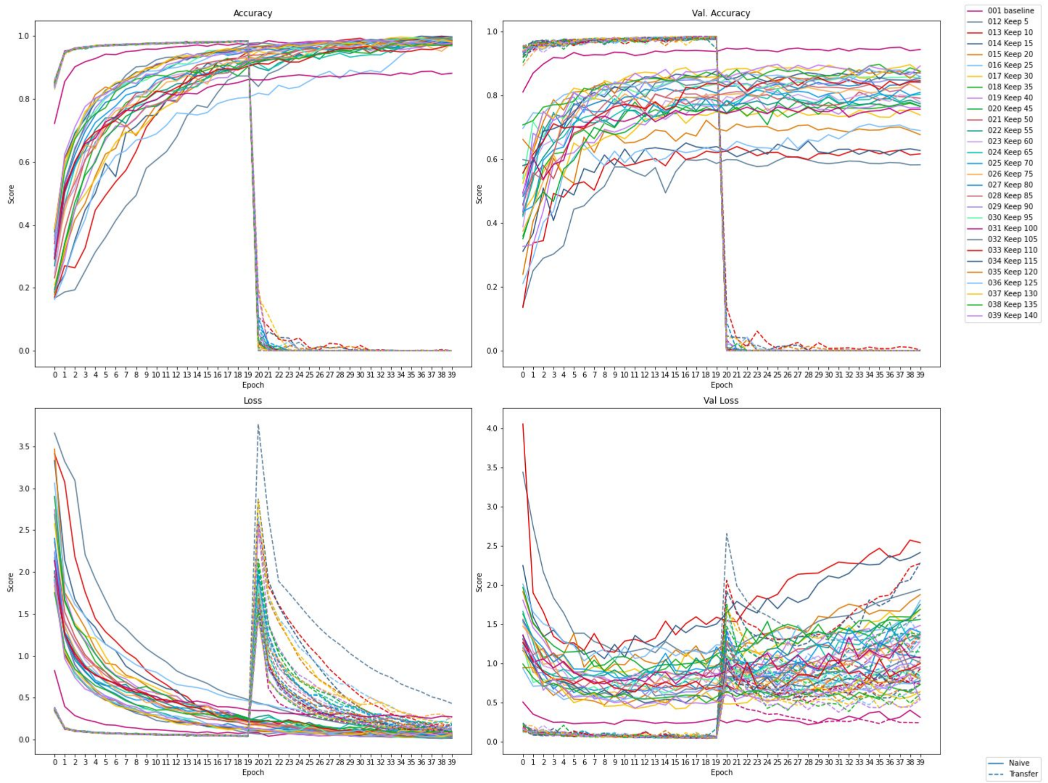
The experiments after 11 looked at keeping *n* samples of each activity in the target training set. The validation and test sets were not restricted.

Overall the trend is that more data improves scores quickly for low values, but starts to converge at values of *n* at roughly 60.

Notably the transfer learning scores largely match the naive scores in all metrics besides accuracy. We are uncertain what is causing these results at this time.

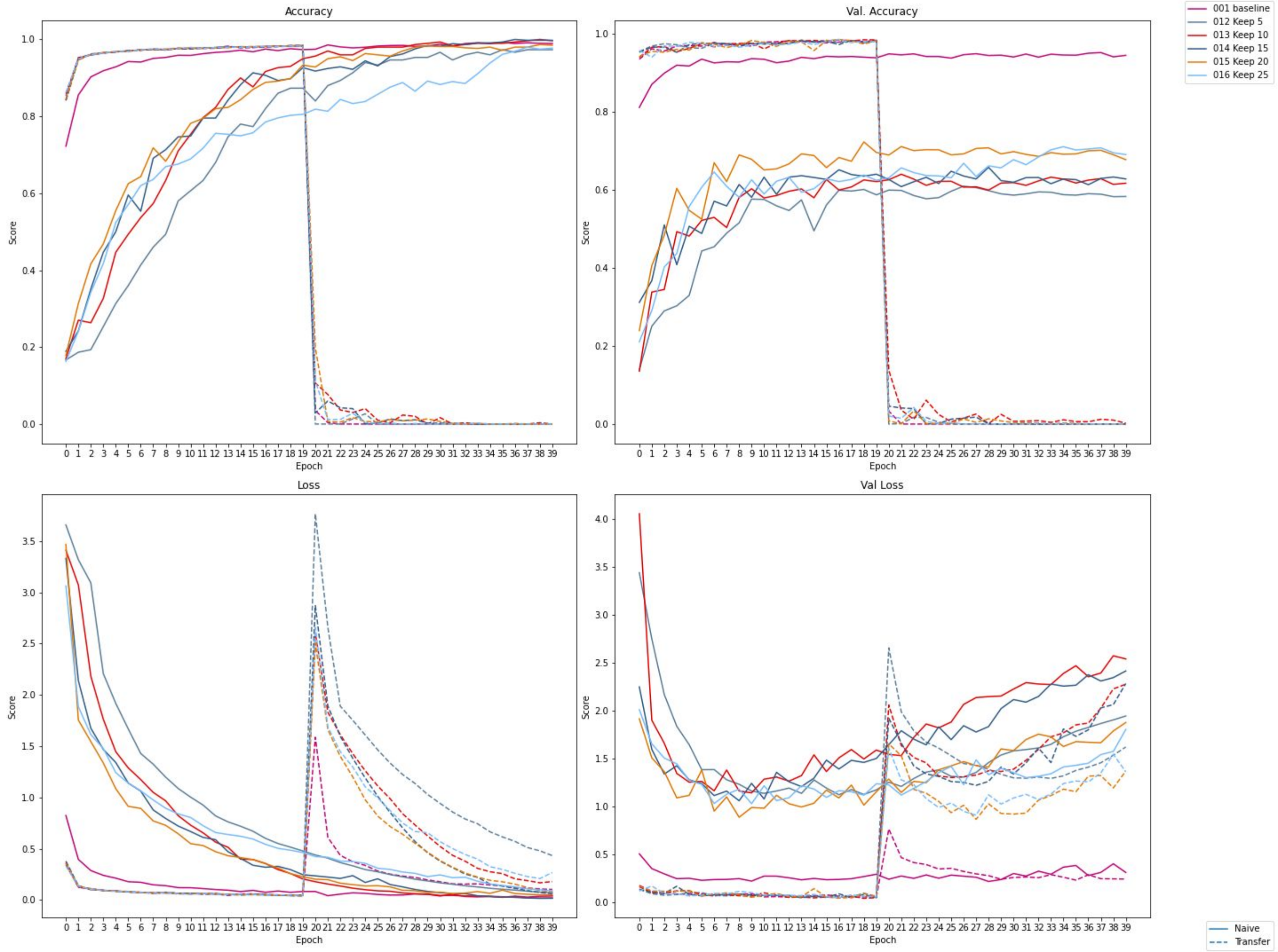


Results: Training Curve

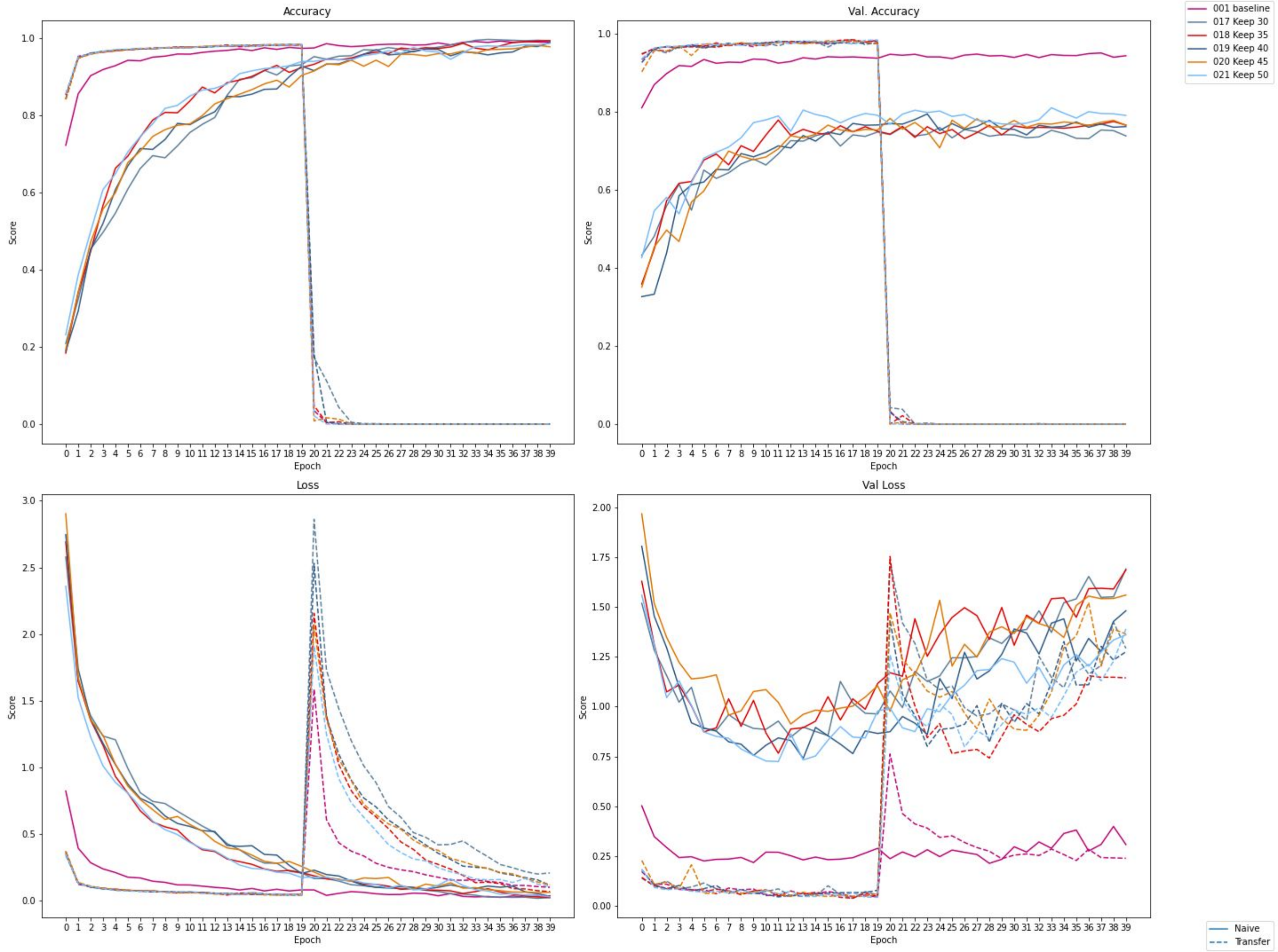


After plotting all learning curves of both the naive and transfer models it became apparent that transfer learning scores became terrible once the transfer learning training started, with both training and validation accuracy almost instantly going to 0 and validation loss rising.

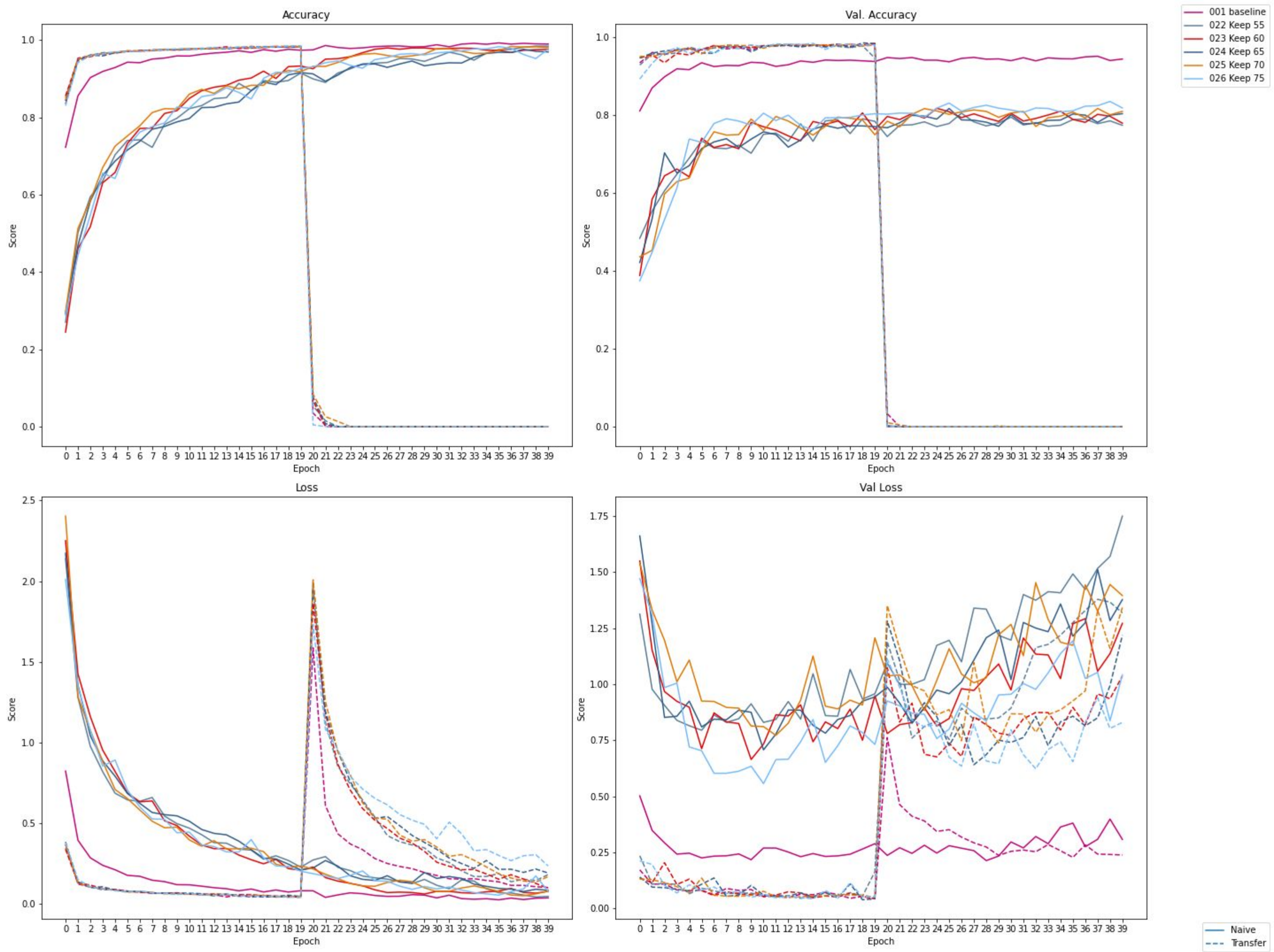
Experiments 1, 12 - 16



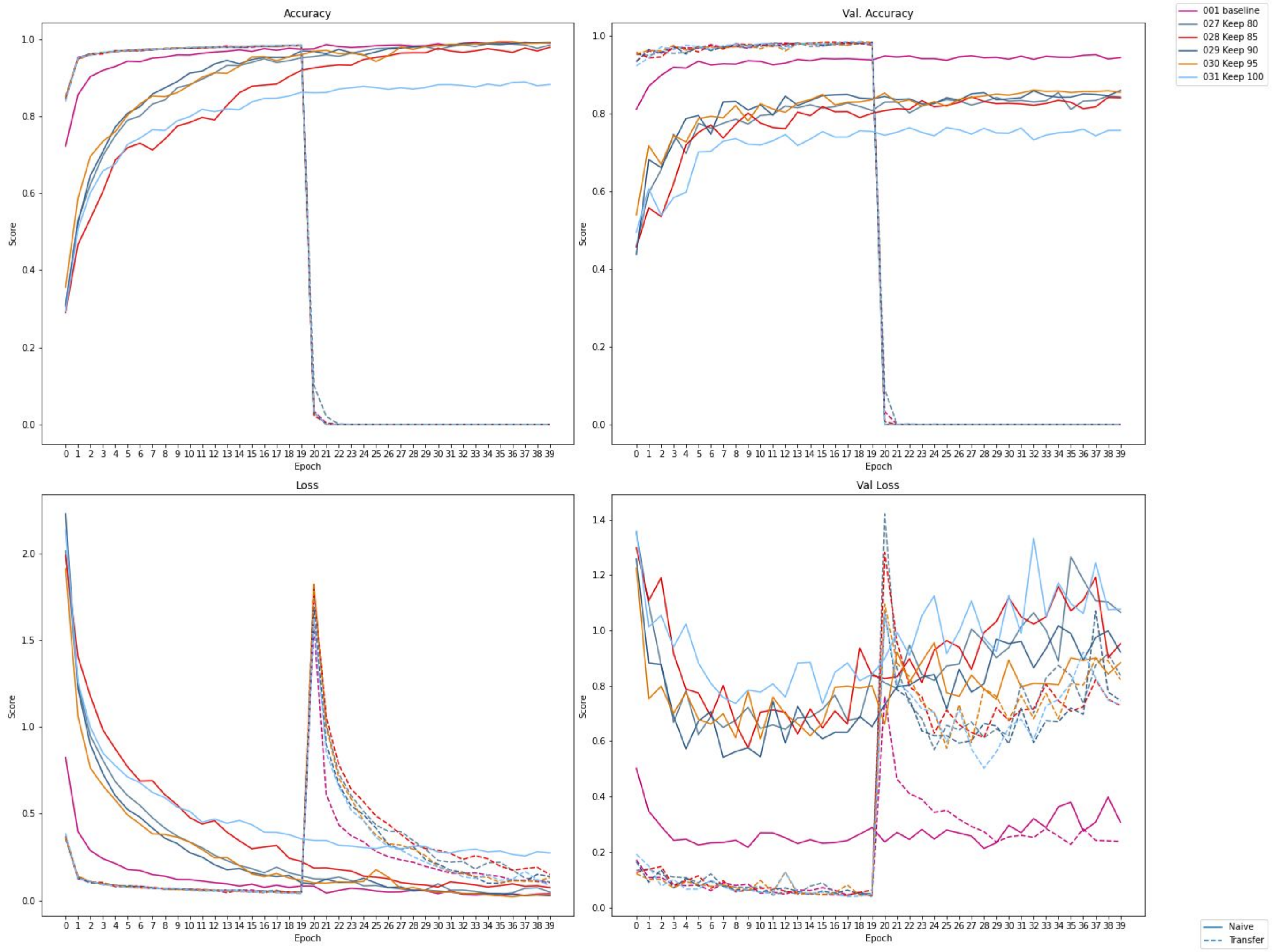
Experiments 1, 17 - 21



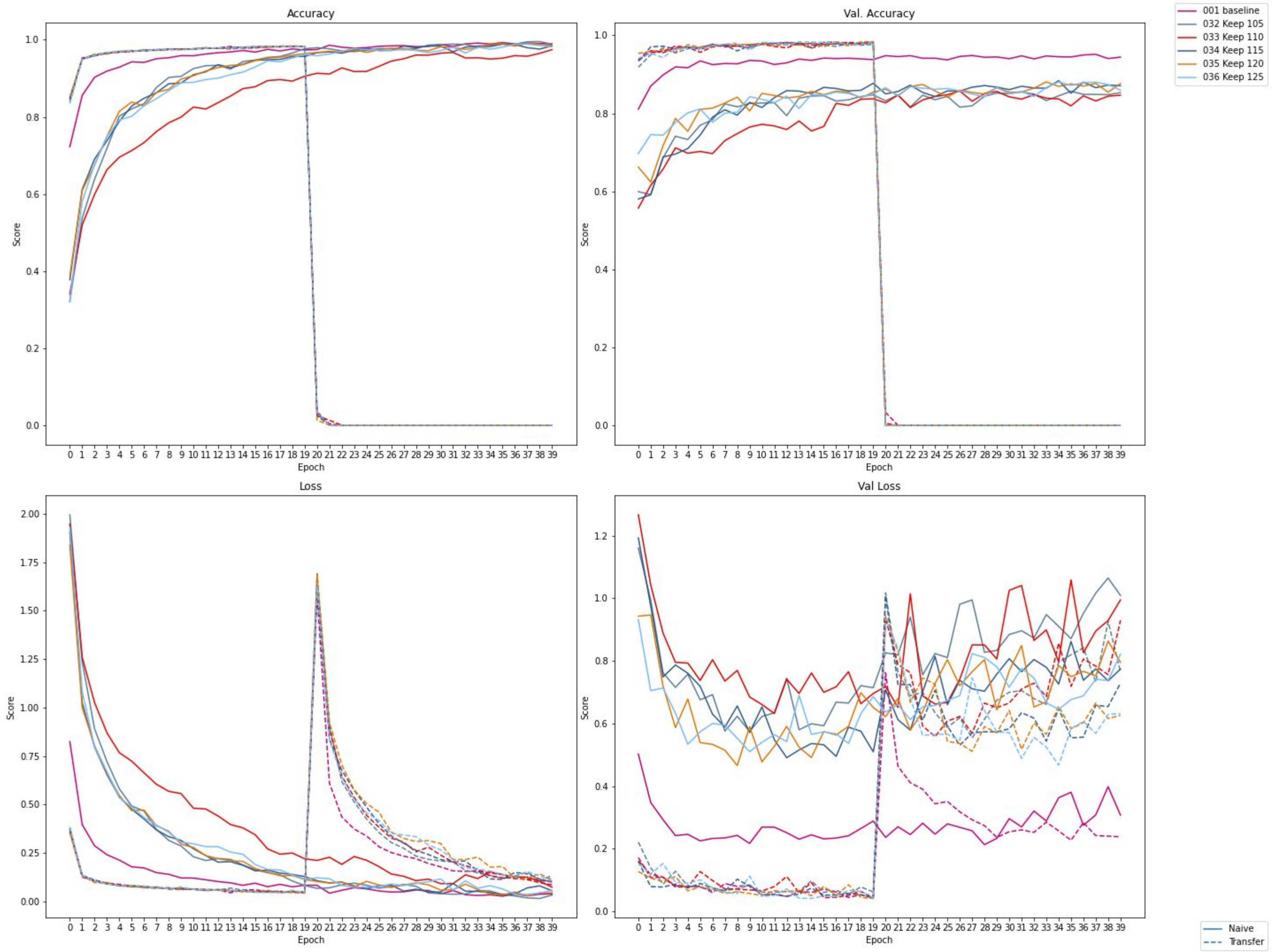
Experiments 1, 22 - 26



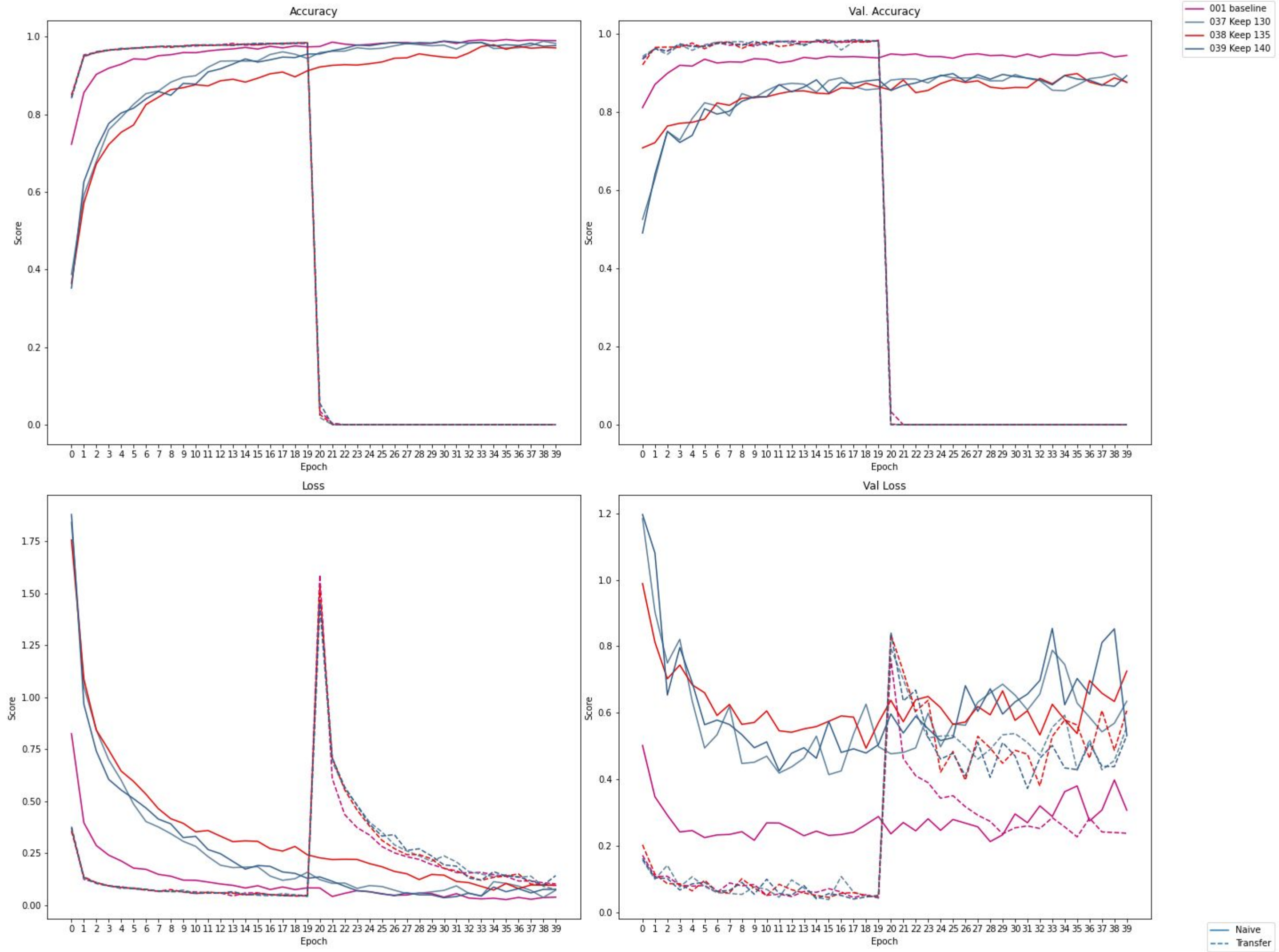
Experiments 1, 27 - 31



Experiments 1, 32 - 36



Experiments 1, 37 - 39



Results: Evaluation

Overall due to the terrible learning curves once transfer learning is enacted, we can say that transfer learning performed poorly in this case. However, the end-of-learning precision, recall, and F1 scores indicate that the transfer learning is indeed working. Further analysis is needed to find the reason for the score discrepancy.

As things stand, transfer learning is possible between datasets. However, the reliability of such learning is dubious.

Archive

Google Drive:
https://drive.google.com/drive/folders/1yNPxni1lCeFX3IWXAt7JjG1VGg554cfD?usp=sharing

Related Projects

HAR Using Deep NN Notebook:

- https://www.kaggle.com/code/euneun000/har-using-deep-nn
- Kaggle notebook of a deep NN applied to the UCI HAR dataset.

Previous Transfer Learning with Gesture Recognition Project Proposal:

- https://docs.google.com/document/d/1KqirKmIku6rEnOwq15qis8jAqeeJV2uS32YsoERjczw/edit
- Project proposal from the Fall 2021 semester of CSCE 5380.

References

An, S., Bhat, G., Gumussoy, S., & Ogras, U. (2020). Transfer Learning for Human Activity Recognition using Representational Analysis of Neural Networks. arXiv preprint arXiv:2012.04479.

Levin, R., Cherepanova, V., Schwarzschild, A., Bansal, A., Bruss, C. B., Goldstein, T., ... & Goldblum, M. (2022). Transfer Learning with Deep Tabular Models. arXiv preprint arXiv:2206.15306.

Pang, J. (2018). Human Activity Recognition Based on Transfer Learning.

Patel, S. (2022, April 15). Deep Transfer Learning for Human Activity Recognition. Medium. Retrieved September 14, 2022, from https://medium.com/@sarjakpatel1999/deep-transfer-learning-for-human-activity-recognition-d291a7731154

Yan, Y., Liao, T., Zhao, J., Wang, J., Ma, L., Lv, W., ... & Wang, L. (2022). Deep transfer learning with graph neural network for sensor-based human activity recognition. arXiv preprint arXiv:2203.07910.