COMSM0045 Applied Deep Learning Report

Milly Pope
Student No.: 2220058

Lucas Chow
Student No.: 2218163

I. Signed Agreement

We agree that all members have contributed to this project (both code and report) in an approximately equal manner.

II. Introduction

Video classification tasks for egocentric videos is a challenging research area within computer science. As humans, it is easy for us to infer context from just a few images. We can fill in what came before and after, keep track of objects that come in and out of frame, and still recognise it's the same scene even when the lighting, framing, or viewpoint shifts. Egocentric videos make that task harder for models.

In this paper we take that challenge and address the issue of pairwise 3-way classification using a Siamese ProgressionNet, using images drawn from the HD-EPIC dataset. More concretely, our aim is: given two single frames, which we denote *anchor* and *comparator*, where each image best represents a step in a recipe, we want to predict their relation. The model must choose one of three labels: the comparator from a later step of the anchor recipe (class 0), an earlier step of the anchor recipe (class 1), or a step from a recipe differing from the anchor recipe (class 2). The anchor and comparator will each be fed into a branch of the Siamese ProgressionNet and produce said classification. Within this paper we explore how we built our base model to produce classifications within an accuracy of $50\% \pm 3\%$ and then go on to discuss extensions we implemented to our base model that improves classification accuracy where we aim for an accuracy of $50\% \pm 1\%$.

III. Related Work

The work we explore within this task encapsulates two areas, Siamese learning for pair matching and self-supervised video methods that use temporal order as a signal.

A Siamese CNN with shared weights is powerful in pair matching when we do not have much labelled data, because we can form lots of training pairs from very few examples, as explored in Sabri et al. [1]. Therefore, we were confident that this architecture gave a good base for two-image problems, and we extend their binary "same/different" setup to a 3-way relation (forward / reverse / different recipe).

Lee et al. [2] pre-train by shuffling four frames and asking the network to sort them back. They introduce an Order Prediction Network that compares all frame pairs before classification, pushing the model to learn temporal/change cues rather than static kitchen context. This was important for motivating our future work and for highlighting where our model may fail given its different architecture; we use simple concatenation rather than any explicit comparison between inputs.

Finally, Misra et al. [3] targets improvements in temporal reasoning, which we came to see as a core difficulty in our network. They design pretext tasks and sampling/augmentations to remove easy background signals, which we discuss in our conclusion as a next step to reduce background bias in our model.

IV. Dataset

All data used within this task originated from the HD-EPIC dataset. This dataset collects kitchen-based egocentric videos with detailed annotations for several ground-truth labels. Nine participants, named P01–P09, produced videos creating recipes. If we have multiple instances of the same recipe, we denote these with instance_M. Specifically for our task, we are interested in the recipe-steps ground-truth labels and have been provided a subset dataset comprised of the following: images that best represent individual steps in a recipe, pre-extracted from videos. As we are comparing two images, we denote the first image the anchor and the second we want to compare against the comparator. We classify as follows: the comparator from a later step of the anchor recipe (class 0), an earlier step of the anchor recipe (class 1), or a step from a recipe differing from the anchor recipe (class 2). Our training dataset consists of 384 images, giving, in principle, up to 384^2 potential ordered pairs; our data is organised by recipe and is accompanied by text descriptions. For validation and test, data is presented instead as pairs of images with label files that allow us to see which index of image belongs to which class. When training we randomly select an anchor and comparator; however, for validation and testing we must evaluate on fixed pairs for comparable results. The class split is equal for validation and test 33/33/33. During training, each pair in a mini-batch is sampled at random, but we use weights $[0.4, 0.4, 0.2]$ for sampling based on classes $[0, 1, 2]$. So the aim is over an epoch we have class split roughly equal to 40/40/20.

V. Siamese ProgressionNet Architecture

The base architecture we implement is the Siamese ProgressionNet as shown in Fig. 1. It is a VGG-style CNN altered to have two branches with shared weights to take two inputs. Each branch encodes the comparator or anchor and consists of eight convolutional layers with 3×3 kernels and padding $= 1$, each followed by Batch Normalization and ReLU; the layers are grouped into pairs to form four convolutional blocks. As annotated in the figure, max-pooling with kernel size 2×2 and stride 2 follows Blocks 1–3, and the final block uses adaptive average pooling to 1×1; the per-block spatial sizes are

$224^2 \to 112^2 \to 56^2 \to 28^2 \to 14^2$, and the channel counts ($C64$, $C128$, $C256$, $C512$), where Ck denotes the number of output channels (filters). Each branch therefore yields a $1 \times 1 \times 512$ tensor that is flattened to a 512-dimensional vector; the two vectors are concatenated to form a 1024-dimensional representation and passed through the two fully connected layers shown at the top (FC $1024 \to 512$, then FC $512 \to 3$). The grey output layer corresponds to the three class logits. Data flow is indicated by the white upward arrows. The purple double-headed arrows denote the Siamese weight tie: corresponding layers in the anchor and comparator branches share parameters.

a) Training: We use `nn.CrossEntropyLoss` as the criterion. The last layer outputs logits; CrossEntropyLoss internally applies softmax so we don't apply it again in the forward pass. The loss is computed as `loss = criterion(logits, labels)`, and predictions use the argmax over logits. Convolution weights are initialized with Kaiming normal and finally inputs are resized to 224×224 RGB (i.e. 3 initial channels) images.

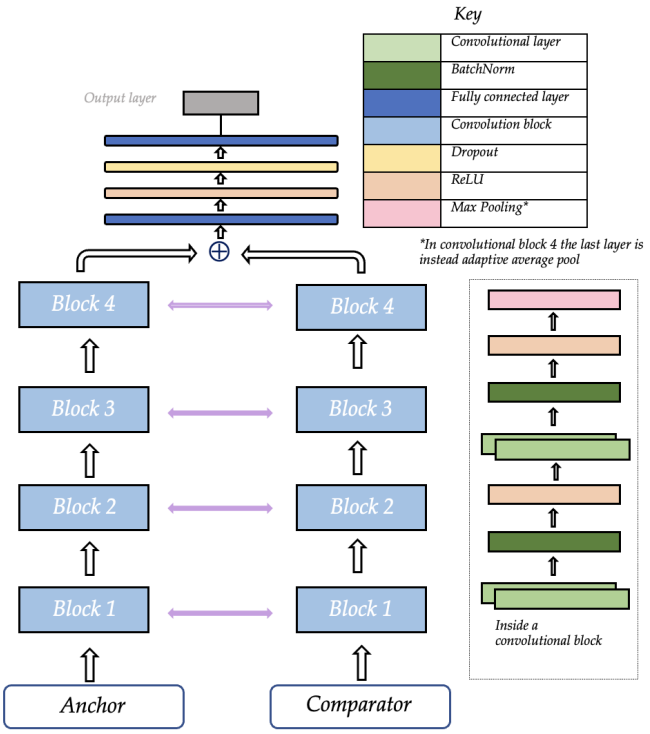

Fig. 1: Siamese ProgressionNet architecture schematic. Left and right branches share weights; features are concatenated and fed to an MLP for three-class prediction.

VI. Implementation Details

We first implemented the model as shown in Fig. 1 and as laid out per the assignment. With configuration (Adam, batch size 32, learning rate 1×10^{-4}, no weight decay). All models were run for 30 epochs with an epoch size 5000. This initial model achieved us an accuracy of 42.39%. Looking at the TensorBoard plots indicated clear overfitting as we had a large generalisation gap and so motivated a grid search over parameters that would enforce regularisation of the model; this included varying weight decay, learning rate, introducing augmentation and using AdamW as an optimiser. The results of this first grid search can be found in Table I.

ID	Acc (%)	Optimizer	LR	WD	Aug
Top-10 runs					
1	46.05	AdamW	1e-4	1e-3	Yes
2	45.47	AdamW	1e-4	1e-4	No
3	45.09	Adam	1e-4	5e-3	Yes
4	44.89	AdamW	1e-4	5e-3	No
5	43.74	Adam	1e-4	5e-3	No
6	42.77	Adam	1e-4	1e-3	Yes
7	42.39	AdamW	5e-4	5e-3	Yes
8	41.62	AdamW	1e-4	5e-3	Yes
9	41.23	Adam	1e-4	1e-4	No
10	41.23	Adam	1e-4	5e-3	Yes
Bottom-5 runs					
30	27.94	Adam	5e-4	5e-3	No
29	29.67	Adam	5e-4	1e-4	Yes
28	30.44	Adam	5e-4	1e-3	Yes
27	31.21	AdamW	5e-4	1e-4	No
26	31.60	Adam	5e-4	5e-3	Yes

TABLE I: Top-10 and bottom-5 out of 30 total runs (middle results omitted for brevity).

We also "pocketed" the best validation checkpoint within each run and used that model to compute test accuracy. We saw the combination of using AdamW with weight decay 1×10^{-3} or 1×10^{-4} performed best. These matched with our intuition of what models may perform better as decoupled L2 style penalties reduce overfitting by shrinking weights and using AdamW gave better performance than Adam under weight decay. The learning rate of 1×10^{-4} consistently outperformed 5×10^{-4}; therefore, going ahead we locked this in as our learning rate. For the specific data augmentation we chose minimal, realistic egocentric variation. Concretely, we used `RandomRotation` ($\pm 5°$) and `RandomHorizontalFlip` ($p = 0.5$), applied identically to both images in the pair. We intentionally did not add vertical flips or large rotations as this was unrealistic for the videos being recorded. We note, interestingly, using augmentation did not provide a clear pattern of better or worse; we concluded this was likely due to the nature of the task and only augmentations that encode the right invariances help, and this is difficult to confidently know which of these are. Our best accuracy of 46.05% was an improvement over our base model but not within the $50\% \pm 1\%$ we aim for.

The TensorBoard plots for our winning model motivated our next steps. The gap between validation and training was smaller but we still saw signs of overfitting. Initially we tried varying dropout but to no improvement. We tested our previous best performing models with dropout values $p \in [0.3, 0.35, 0.4, 0.45]$ as we were still seeing overfitting; however, this achieved accuracy results lower than with a dropout value of 0.5. Inspecting confusion matrices produced as well as TensorBoard plots we ascertained this was due to this simply being a weaker form

Label smoothing ε	Test Acc (%)	Test Loss
0.05	47.78	1.28528
0.20	46.82	1.20703
0.15	46.44	1.21992
0.10	43.74	1.45001

TABLE II: Label smoothing results.

of regularisation; confusion matrices saw worse generalisation and lower per class accuracy with the model much more often defaulting to classes 0/1. Keeping dropout at 0.5 gave a better bias–variance trade-off and so we moved onto label smoothing. This regularisation technique achieved much more promising results. With our previous best performing model we explored values of $\epsilon \in [0.05, 0.1, 0.15, 0.2]$. This achieved the results in Table II. Therefore our base model within our desired accuracy used AdamW as an optimizer, a learning rate of 1×10^{-4}, weight decay parameter of 1×10^{-3}, label smoothing with value 0.05 and finally with augmented data during training.

VII. Replicating Quantitative Results

Our final results for accuracy can be found in Table III, and the confusion matrix produced on the test set in Fig. 2. Our matrix shows

$$\begin{aligned} \text{Recall: } & 0 \to 68.2\%, \quad 1 \to 67.1\%, \quad 2 \to 8.1\% \\ \text{Precision: } & 0 \to 47.6\%, \quad 1 \to 46.6\%, \quad 2 \to 63.6\% \end{aligned}$$

We are seeing the model sparsely predict class 2 although when it does its usually right and a symmetric confusion between classes 0 and 1 which is to be expected as they are visually close. What this model does well is only predict class two when its confident - we see very few false positives and is able to separate classes 0/1 reasonably well. However what our model fails in is predicting class 2, our network is hesitant to predict that class so we see good precision but bad recall. We reason the following patterns arose due to the subtle differences in classes 0/1 therefore expect the model to struggle differentiating them. In addition to this our model is not as good as predicting class 2 as while the training set includes category 2 pairs from different kitchens making it easy for the model to identify different recipes as they are from different kitchens, the category 2 pairs in the test set are from the same kitchen, so a model trained to discriminate based on different kitchens will assume these pairs are from the same recipe, as they are in the same kitchen, neglecting features like ingredients or utensils.

TABLE III: Test performance on recipe progress classification.

Model	Accuracy (%)	Notes
Siamese ProgressionNet (ours)	47.78	Baseline reproduction

$$\begin{pmatrix} 118 & 51 & 4 \\ 53 & 116 & 4 \\ 77 & 82 & 14 \end{pmatrix}$$

Fig. 2: Confusion matrix.

VIII. Training Curves

Our accuracy curves are shown in Figs. 3 and 4. We see a generalisation gap $\approx 0.25 - 0.3$ which, although it could be smaller, is to be expected and is a big improvement on initial models. We see validation loss going upwards whilst validation accuracy plateaus as later in the training the model gets more confident and so for the validation samples it incorrectly predicts it does so confidently; therefore, their loss contribution increases, leading to validation loss increasing even though the validation accuracy isn't changing as much – we're seeing overconfidence. This helps explain why introducing label smoothing was able to produce our best model as it aims to dampen this.

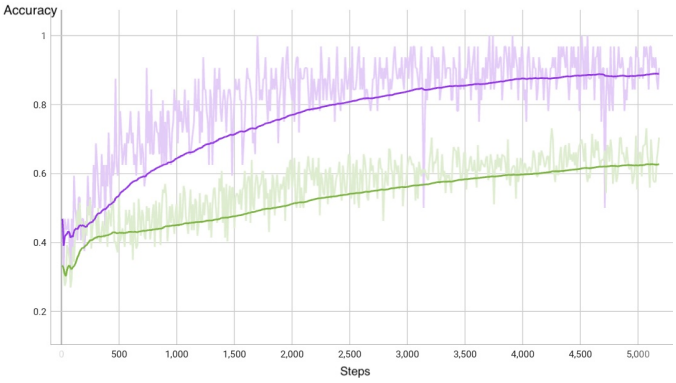

Fig. 3: Accuracy across epochs; Training: Purple, Validation: Green.

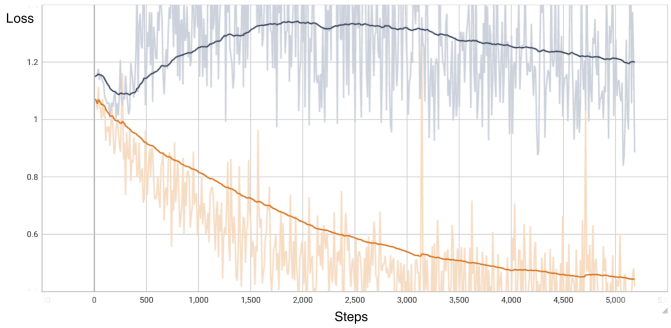

Fig. 4: Loss across epochs; Training: Orange, Validation: Grey.

IX. Qualitative Results

In the above sections we concluded our model was conservative in predicting class two and the prediction of this class is where much of the inaccuracy of our model lay (i.e., identifying different recipes). Looking into the correct and

incorrect prediction gives us a good intuition as to why. In the correct prediction we clearly see different angle shots, backgrounds and utensils—strong indicators of different recipe. In our training data, class-two pairs often came from different kitchens, so these big context changes were common and usually meant "different recipe". The model became dependent on those cues; when they are present it fires class two (and is usually right, $\text{precision}_2 \approx 63.6\%$), but when they are not there is not much room for correctly identifying different recipes.

Evidence of this can be seen in the incorrectly classified image pairs: they contain the same backgrounds, utensils and angles of shots. On the test set many class-two pairs are from the same kitchen, so those easy context signals are missing. This is what is creating our low recall for class two ($\text{recall}_2 \approx 8.1\%$)—our model avoids class two and instead picks classes 0/1 for different but similar-looking recipes, which also leads to the symmetric $0 \leftrightarrow 1$ confusions we observed.

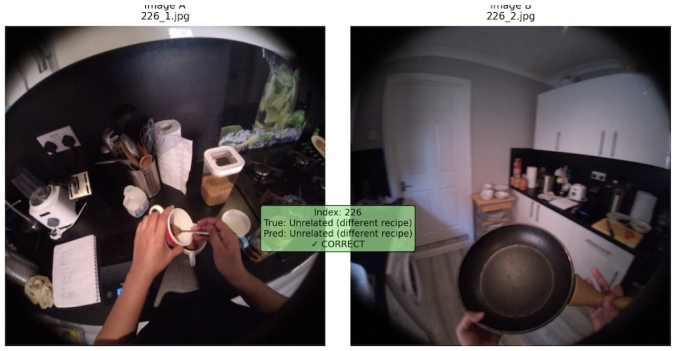

Fig. 5: Correctly classified pairs.

Fig. 6: Incorrectly classified pair 1.

Fig. 7: Incorrectly classified pair 2.

X. Extension

We move forward in discussing how extending our base model achieved us our desired accuracy of $50\% \pm 1\%$. It was clear to us that the main issues in our model lay in class 2 predictions. Therefore we needed to fine tune the model so that it performed better for this task. To do this we wanted to expose our model to many more class 2 images, giving it more data and thus allowing it to learn how to predict class 2 better. To do this we edited the data loader which, as previously discussed, sampled from classes $[0, 1, 2]$ with respective probabilities/weights $[0.4, 0.4, 0.2]$. We altered these weights to favour class 2 more heavily and took our best model and continued training it using an altered data loader (which sampled with the altered weightings) for a further 5 epochs. Once again we pocketed the best model based on validation accuracy each epoch.

XI. Extension Results

Our best model achieved a test accuracy of 51.65% on epoch 4 with the data loader sampling classes via the following probability distribution $[0.25, 0.25, 0.5]$. Our main aim with this extension was to improve the class accuracy for class two although we aimed for an overall improvement in accuracy too. Our earlier model was conservative on class 2: it predicted "different recipe" only in obvious cases, we saw high precision but very low recall ($\text{precision}_2 \approx 63.6\%$, $\text{recall}_2 \approx 8.1\%$). After fine-tuning by training more on class 2, the confusion matrix in Fig. 8 shows that the model no longer requires extreme evidence to call class 2: recall_2 rises to $73/173 \approx 42.2\%$. This directly addresses the issue of over-confidence we were seeing. Although we see precision_2 drops (to $73/147 \approx 49.7\%$) because we now predict class 2 more often, overall accuracy improves (from $47.8\% \rightarrow 51.6\%$). Classes 0/1 give up some recall ($\approx 68\% \rightarrow \approx 56\%$), yet their precision improves (to $\approx 53\%$ and $\approx 52\%$), indicating a better-balanced operating point.

$$\begin{pmatrix} 97 & 39 & 37 \\ 38 & 98 & 37 \\ 47 & 53 & 73 \end{pmatrix}$$

Fig. 8: Confusion matrix.

TABLE IV: Extension Accuracies.

Variant	Accuracy (%)	Aim
Baseline	47.78	$50\% \pm 3\%$
Baseline + Extension	51.64	$50\% \pm 1\%$

XII. Conclusion and Future Work

Our initial Siamese ProgressionNet saw a quick rise to a high training accuracy whilst validation accuracy plateaued around 0.54. This was expected given the nature of the task. The images were taken from egocentric and visually repetitive data and many of the different recipe classification pairs came from different kitchens, meaning background/kitchen identity became an easy shortcut for this classification—especially with 30 epochs and 5000 steps per epoch and no regularisation on the model. It became conservative on class 2 predictions and showed symmetric class 0/1 confusions where visual differences were subtle. Our network essentially learned the easy context and stayed under-confident about class 2 whenever context wasn't an obvious cue for different recipes.

We aimed to fix this "too easy to memorise" issue by enforcing regularisation using AdamW with weight decay, and light augmentation to mitigate overfitting. Introducing label smoothing helped with later overconfidence; however, we still saw the model scarcely predicting class 2.

For our extension we targeted this error pattern specifically by increasing the model's exposure to class 2 during further training (class-weighted cross-entropy / light oversampling), which directly addressed the under-calling of class 2. We saw class 2 recall increase from $8.1\% \rightarrow 42.2\%$. The trade-off was a decrease in class 2 precision; however, overall accuracy improved (from $47.8\% \rightarrow 51.6\%$).

Our main takeaways from this task were that the necessary regularisation impositions arose from the dataset letting background serve as an easy learning shortcut. We learned that evaluating the confusion matrix signals any class under-calling, and that this can be fixed by feeding the model more data/weighting for the under-called class during training.

1) Future work: The main limitation we found within this task was that the test dataset only included a single participant; therefore, dependencies learned in training (which used multiple kitchens/participants) did not translate as well when testing the model. In future, to improve our model we would train specifically on pairs from different recipes with similar backgrounds, angles and lighting, forcing it to reliably learn recipe-level cues (ingredients/utensils/state) rather than differentiating by background, as we observed in IX. As explored in [3] we would aim to train our model on signals such as utensil changes or poses rather than kitchen context; we would also consider augmentations that specifically address the issue of learning obvious background dependencies such as spatial jittering or channel splitting. Finally, ideas within [2] motivate us to explore using a cross-frame interaction head before the final 3-way classifier instead of just concatenation to allow the network to explicitly compare the anchor and comparator.

References

[1] N. I. Ahmad Sabri and S. Setumin, "One-Shot Learning for Facial Sketch Recognition using the Siamese Convolutional Neural Network," in *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Apr. 2021, pp. 307–312. [Online]. Available: https://ieeexplore.ieee.org/document/9431773/

[2] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised Representation Learning by Sorting Sequences," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 667–676, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/document/8237341/

[3] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification," Jul. 2016, arXiv:1603.08561 [cs]. [Online]. Available: http://arxiv.org/abs/1603.08561