COMSM0045 Coursework: Exploring Siamese CNNs Using the HD-EPIC Dataset

os22128 — Nicholas Harrod

xd22898 — Valentina Velasco

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

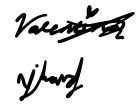

Fig. 1. Signatures for Nicholas Harrod and Valentina Velasco

I. Introduction

Within the field of deep learning for computer vision, there are a plethora of different challenges to face when it comes to comparing videos and the frames from them. In egocentric videos, this challenge further requires understanding the underlying human task and its position within a sequence. To address this, we propose a solution utilising a Siamese Convolutional Neural Net (CNN) model to compare recipe images from the HD-EPIC Dataset [1] and further improve performance through data augmentation and modified activation functions.

The structure of this report is as follows. Section II reviews related work. Section III describes the HD-EPIC dataset and the pairing scheme used. Section IV presents the Siamese Progression Net architecture. Section V outlines our implementation details and hyperparameter selections. Section VI reports our quantitative results, followed by training curve analysis in Section VII. Section VIII provides qualitative examples, while Section IX discusses improvements and extensions. Finally, Section X concludes the report and highlights directions for future work.

II. Related Work

A. *Cooking Activity Recognition in Egocentric Videos with a Hand Mask Image Branch in the Multi-stream CNN*

Michibata et al. [2] propose a technique for activity recognition in cooking videos that focuses on separate streams of the video to recognise the user's actions. Even though they had a separate goal to ours, their use of image segmentation offers a novel perspective on how models process and analyse egocentric images around cooking activities.

B. *Dual-Path Siamese CNN for Hyper spectral Image Classification With Limited Training Samples*

Huang and Chen explore different hyper-parameters, modifications and training techniques for Siamese CNNS and compare traditional CNN models with dual path Siamese architectures [3]. Although their work targets a very specific dataset, their results offer valuable benchmarks demonstrating the benefits of using a Siamese CNN and provide insight on which hyperparameters to tune and focus on, as well as using dropout - an extension we toyed with during development.

C. *Face recognition using CNN and Siamese network*

Kumar et al. propose a facial recognition model that, through experiments similar to those in the previous work, demonstrates the advantages of using Leaky ReLU over standard ReLU [4]. The strong performance reported in their study led us to adopt Leaky ReLU in our own model, as discussed in Section IX.

III. Dataset

The HD-EPIC dataset contains egocentric video recordings of different individuals carrying out various recipe preparation tasks. Whilst the wider dataset has a large amount of multimodal data points such as 3D kitchen models and "hand-masks", our process focused on a specific stream of the dataset - pairs of images that are either from the same recipe video or different. These frames are taken from explicitly defined recipe "steps" that represent key moments in the cooking process.

Each image pair is divided into an anchor and comparator (A, C) and is assigned to one of three classes: 0 — same recipe, comparator after anchor; 1 — same recipe, comparator before anchor; 2 — different recipes. The dataset is organised such that each pair has a unique ID and a corresponding entry in a *labels.txt* file indicating its class. The split was as follows:

2000 train pairs, 189 validation pairs and 159 test pairs

The number of training pairs was controlled by a hyperparameter called *epoch_size* which defined how many pairs from the 66 available recipes were fed to the model per epoch.

IV. Siamese ProgressionNet Architecture

Our model is based on a dual-stream Convolutional Neural Network (CNN) architecture known as a Siamese network. It processes an anchor and comparator image (the A and C of the dataset) through two identical, weight-sharing branches. The resulting feature representations are then fused and passed to a fully connected head, whose outputs are trained using cross-entropy loss.

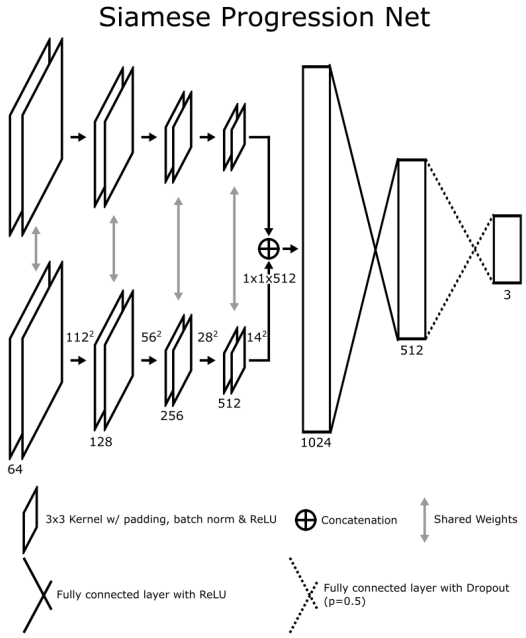

Fig. 2. Digram of the Siamese Progession Net Architecture

1) Feature Extraction (Siamese Branches): The input is processed by two parallel convolutional branches. Each branch acts as a feature extractor and the weights are shared between branches, ensuring that both inputs are mapped into the same feature space.

The progression of the feature maps in each branch is as follows:

- **Stage 0:** Pre-Processing - The input of size $512 \times 512 \times 3$ is narrowed to $224 \times 224 \times 3$ to fit the model.
- **Stage 1:** Convolution Block 1 - The input is processed to produce a feature map of size 112×112 with 64 channels.
- **Stage 2:** Convolution Block 2 - The spatial dimension is halved to 56×56, while the channel depth increases to 128.
- **Stage 3:** Convolution Block 3 - The spatial dimension reduces to 28×28 with 256 channels.
- **Stage 4:** Convolution layer 4 - This stage yields a 14×14 feature map with 512 channels.

Each convolutional block consists of two convolution layers utilising 3×3 kernels with padding, followed by Batch Normalization, a Rectified Linear Unit (ReLU) activation function and a Pooling layer in which the model uses MaxPool.

2) Fusion and Classification: Following the final convolutional block, there is an AveragePool layer and the output feature vectors ($1 \times 1 \times 512$) from both branches are merged via concatenation ($\oplus$). This results in a combined feature vector of size 1024.

This fused vector is fed into a Fully Connected (FC) network with the following structure:

1) A dense layer with 1024 units.
2) A dense layer with 512 units. Both of these layers utilize ReLU activation.
3) A final output layer with 3 units.

To reduce overfitting, dropout with a probability of $p = 0.5$ is applied to the connections before the final output layer.

V. Implementation Details

A. Hyperparameter Search and Final Configuration

To identify an effective training configuration for the three-way recipe comparison task, we carried out hyperparameter search over the learning rate, batch size, epoch size, and the number of epochs in order to maximise test accuracy.

a) Batch Size.: All experiments were initially run with batch sizes of 16 and 32. A batch size of 16 led to weaker feature extraction and lower accuracy, whereas a batch size of 32 produced significantly higher validation accuracy and better generalisation to the test set.

b) Learning Rate.: We tested learning rates in the range 5×10^{-5} to 5×10^{-4}. Lower learning rates (e.g. 5×10^{-5} or 1×10^{-4}) led to slow convergence where the model was unable to make meaningful progress. High learning rates (4×10^{-4} to 5×10^{-4}) induced rapid overfitting: batch accuracy rose above 95% consistently, while validation and test accuracy degraded. The most stable learning behaviour emerged around 2×10^{-4} to 3×10^{-4}.

c) Epoch Size.: The number of batches sampled per epoch ("epoch size") was varied between 800 and 3000. Smaller sizes provided a mild regularising effect but less stable training behaviour. Larger sizes improved representation learning but increased susceptibility to overfitting at higher learning rates. An epoch size of 2000 provided the best balance between stability and generalisation.

B. Training Schedule

The model was trained for 20–60 epochs during tuning. Runs with fewer than 30 epochs underfit, whereas 60-epoch runs tended to overfit unless paired with a conservative learning rate. Training for 50 epochs provided stable learning without excessive memorisation. The learning rate remained constant throughout training, as this provided more stable convergence than using a scheduler or weight decay. No early stopping was used for the final configuration, as validation loss occurred very early and did not correspond reliably to improvements in validation/test accuracy. Early stopping based on validation accuracy also tended to underfit so would decrease the test accuracy.

VI. Replicating Quantitative Results

A. Optimal Hyperparameters

The final hyperparameters for our base model that produced the best generalisation performance were

- **Epochs:** 50
- **Learning rate:** 2.5×10^{-4}
- **Batch size:** 32
- **Epoch size:** 2000 steps per epoch
- **Optimizer:** Adam
- **Weight decay:** none
- **Early stopping:** disabled

B. Resulting Performance

With this configuration, the model achieved the highest performance during the tuning process:

- **Training accuracy: 90-95%** in the final epochs
- **Validation accuracy: 50–60%**
- **Test accuracy: 50.67%**
- **Confusion matrix:**

$$\begin{bmatrix} 131 & 40 & 2 \\ 48 & 121 & 4 \\ 87 & 74 & 12 \end{bmatrix}$$

The confusion matrices in this report use a consistent layout, with predicted classes on the horizontal axis and actual classes on the vertical axis. For class 0, the model correctly classified 131 instances and misclassified 42 (40 as class 1 and 2 as class 2). Ideally, all entries would appear on the diagonal, with zeros in the off-diagonal positions.

While the model performs reasonably well on Class 0 and Class 1, it performs very poorly on Class 2, with only 12 correct predictions and a high number of misclassification's into the other two classes. This indicates that the model fails to learn a discriminative representation for Class 2 under the current setup. Therefore, our improvements are aimed at addressing this weakness and improving the test accuracy.

VII. Qualitative Results

In this section, we present qualitative examples showing where the model performs well and where it struggles. These examples include comparisons between the model's predicted output (*pred*) and the corresponding ground-truth label (*label*).

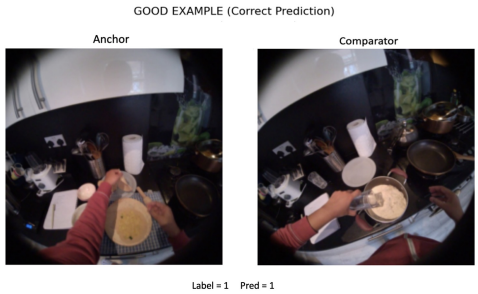

Fig. 3. Successful example where the model correctly predicts the label, *pred = 1*).

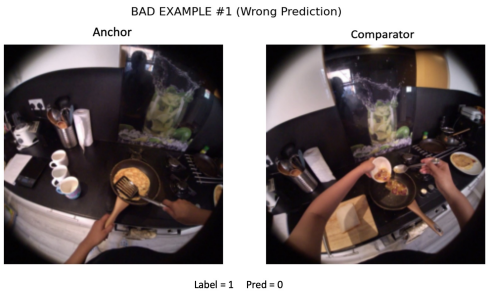

Fig. 4. Challenging example where the model incorrectly predicts the label (*label = 1, pred = 0*).

Fig. 5. Challenging example where the model incorrectly predicts the label (*label = 2, pred = 0*).

From the examples, we observe several consistent patterns in the model's behaviour. In the figure 4, although the correct label is *1* indicating the comparator occurs before the anchor, the model predicts the opposite. This is because the model identifies the fully visible omelette in the comparator but fails to recognise the cropped omelette visible in the background of the anchor. This indicates a limitation in the model's ability to interpret partial objects and perhaps meaningful temporal cues such as the removal of the chopping board. In the second challenging case (Figure 5), the model struggles when the anchor and comparator images are both captured in the same kitchen environment, under nearly identical lighting conditions and involve visually similar actions - suggesting the model has difficulty distinguishing between different recipes when the visual context remains highly consistent. The successful example in figure 3 on the other hand demonstrates that the model can correctly infer the temporal/semantic relationships when the visual cues are sufficiently clear.

VIII. Improvements (Extensions)

A. Leaky ReLU

Kumar et al. [4] observed that using Leaky ReLU led to a "significant improvement in the model's performance." To incorporate this benefit, we replaced the standard ReLU activation in our architecture with Leaky ReLU.

Rectified Linear Unit (ReLU) works by taking the input value if it is positive and 0 otherwise, effectively removing all negative values from the system. Leaky ReLU instead multiplies the value of any negative input by a coefficient α less than 1 to still penalise negative values but retain some of their information.

$$\text{ReLU}(x) = \max(0, x)$$

$$\text{LeakyReLU}(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$$

$$\begin{bmatrix} 133 & 34 & 6 \\ 47 & 120 & 6 \\ 80 & 66 & 27 \end{bmatrix}$$

The use of Leaky ReLU with an α value of 0.01 increased

Fig. 6. Image pairs labelled as class 2 in the validation and test sets

Fig. 7. The result of cropping 40% from the height and 20% from the width

our model's accuracy by **3.28%**, resulting in a final test performance of **53.95%**—a substantial improvement over the baseline architecture. However, the confusion matrix shows that the model continues to struggle with class 2, which remains under-represented in both correct and incorrect predictions. This is an issue we attempted to address in our next improvement to the model. One aspect worth noting is that Leaky ReLU introduces slightly higher computational complexity than standard ReLU, which in turn slightly increases the model's environmental footprint. However, because this added cost is minimal—especially when compared to heavier activation functions such as sigmoid or tanh—we decided to keep Leaky ReLU in our architecture.

B. Data augmentation

To address the uneven distribution of per-class accuracy, we examined the model's inputs more closely and analysed how the network was making its predictions. In particular, we focused on image pairs classified as class 2 in both the validation and test sets. As displayed in Figure 6

Although these examples come from two distinct recipes, the validation images were captured in completely different environments, while the test images were taken in the same kitchen under identical lighting conditions. After identifying this, we introduced two data-augmentation strategies before the images were fed into the network..

1) Greyscale: Because the training set contained class-2 pairs from entirely different recipes prepared in different kitchens, our aim was to reduce the impact of colour on the model. This helps limit overfitting by encouraging the network to learn more about the shapes, textures, and spatial features - rather than background colours or lighting. However, since colour is still a good indicator of different ingredients and carries important semantic information (for instance, a white onion and an apple share a similar shape but differ in colour), we applied the greyscale augmentation with only a 10% probability. This adjustment had an immediate positive effect on the per-class accuracy, as shown in the confusion matrix below.

$$\begin{bmatrix} 113 & 25 & 35 \\ 52 & 85 & 36 \\ 68 & 43 & 62 \end{bmatrix}$$

50.10 % This reduction in overfitting is further supported by the improved train–validation accuracy curve shown which combines both leaky ReLU and greyscale. Figure 10.

2) Cropping: To further improve data augmentation, we cropped the image in order to focus it on specific parts of the data. As illustrated in the images below, the key features of the cooking activity-and therefore the elements that distinguish one recipe from another are focused in the lower-central portion of the image. This crop is shown in Figure 7. By restricting the model's attention to this region, our aim was to reduce the influence of the background and encourage the network to learn only from the main activity being performed. This approach mirrors the strategy used by Michitbata et al., who applied hand segmentation to isolate cooking activity in their work.

When combined with greyscale this turned out to be a misstep as seen in the confusion matrix below:

$$\begin{bmatrix} 131 & 42 & 0 \\ 38 & 135 & 0 \\ 76 & 97 & 0 \end{bmatrix}$$

Despite a drop in the per class 2 accuracy, the performance on the test set still increased by **0.58%** leading to a test accuracy of **51.25%**. Whilst this improved our accuracy compared to the base model, the issue of class accuracy distribution became even more apparent-the model was now predicting classes 0 and 1 with **77%** accuracy, while almost entirely neglecting class 2.

C. Combining Improvements

These improvements shine a light into the nature of quantitative results against qualitative ones. Going off metrics alone, all of the implemented extensions to the model lead to better performance than the base model (in terms of test accuracy) but when analysing other statistics, it becomes clear that some extensions should be discarded.

Leaky ReLU by itself gave the best scores for accuracy but had drawbacks when it came to correctly predicting class 2. Greyscale with Leaky ReLU had a lower accuracy score but far more correct class 2 predictions. Cropping on top of all of this had a similar accuracy score to Greyscale but the harshest class distribution seen so far, with not a single pair being predicted in class 2.

Of these improvements, we concluded that the best model going forward for practical use and further improvement would be leaky ReLU with greyscale since per-class accuracy is a very important metric.

IX. Training Curves

A. Base Model

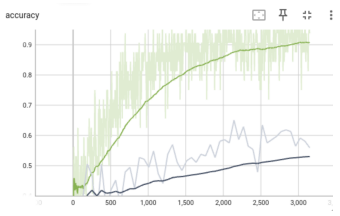

Fig. 8. Training and validation accuracy curves for the base model.

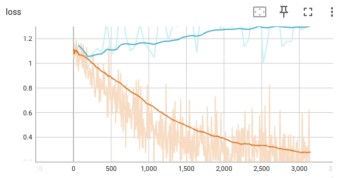

Fig. 9. Training and validation loss curves for the base model.

The base model (trained without augmentation, weight decay, or modified activation functions) shows clear signs of mild overfitting, as shown in Figure 8. The training accuracy increases steadily and approaches approximately **94%**, while the validation accuracy plateaus around **55–60%**. This widening gap indicates that the model is learning class-specific patterns present in the training data that do not generalise well to unseen samples. Nevertheless, the validation curve does not collapse; instead, it improves gradually across the epochs, suggesting that the model still exhibits moderate generalisation rather than severe overfitting. In addition to the accuracy trends, the validation loss gradually increases over time (Figure 9), indicating that the model becomes less confident on unseen data. Although this rise is much slower than the rapid decrease in training loss, it still reflects the limited generalisation capacity observed earlier.

B. Improved Model

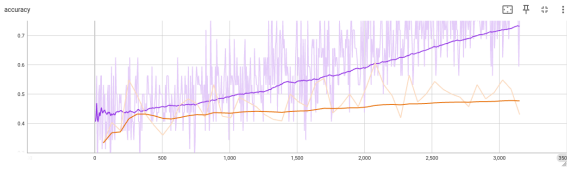

Fig. 10. Training and validation accuracy curves for the improved model.

Compared to the base model (Figure 8), the gap between training and validation accuracy is substantially smaller as shown in (Figure 10), and the point at which the curves begin to diverge occurs much later in training. This indicates that the model is learning more generalisable representations rather than rapidly overfitting to the training data. Notably,

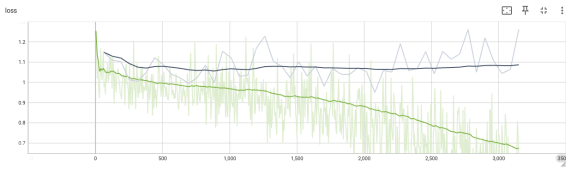

Fig. 11. Training and validation loss curves for the improved model.

the training accuracy in the improved model rises only to around **70%** by the final epoch, compared to almost **90%** in the base model, further demonstrating that the model is no longer memorising the training set. A similar trend is visible in the loss curves: the improved model's training and validation losses (Figure 11) remain much closer together than in the base configuration (Figure 9), and the validation loss increases more slowly over time. Together, these curves demonstrate that the applied modifications significantly enhance the model's stability and reduce overfitting, leading to better generalisation performance.

X. Conclusion and Future Work

Using the base model, we achieved a high classification accuracy of **50.67%** and achieved a peak accuracy of **53.95%** after adding some improvements involving Leaky ReLU. We also refined the per class accuracy of the model making it more applicable in real deep learning scenarios through exploration of extensions involving greyscale data augmentation.

We were surprised that the cropping data augmentation actually reduced generalisation instead of improving it and in the future, a good place to build upon this would be to experiment with different cropped formats and shapes. However, as a result of having a model that achieves **77%** accuracy for class 0 and 1, this model could be included in a multi-headed network format that includes a separate network used to distinguish between class 2 pairs and class [0-1] pairs, utilising our model as the classifier for the [0-1] pairs. This could give results soaring far past what we achieved due to the high accuracy of our model.

References

[1] T. Perrett et al., "HD-EPIC: A Highly-Detailed Egocentric Video Dataset," In: Proc. CVPR, 2025.

[2] Michibata, S., Inoue, K., Yoshioka, M., Hashimoto, A.: Cooking activity recognition in egocentric videos with a hand mask image branch in the multi-stream CNN. In: Proceedings of ACM CEA 2020, pp. 35–40

[3] L. Huang and Y. Chen, "Dual-Path Siamese CNN for Hyperspectral Image Classification With Limited Training Samples," in IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 3, pp. 518-522, March 2021

[4] Face recognition using cnn and siamese network Ranjeeth Kumar C., Saranya N., Priyadharsini M., E D.G., M K.R.Measurement Sensors 2023