Of Mice and Men : CS 7643

Emilio Moyers emoyers3
Diana Lomelin dbustos3
Jonathan Barton jbarton43
Andres Menendez amenendez6

Abstract

The brain is an immensely complex system that acts accordingly when confronted with different scenarios. These actions can be categorically defined as behaviors and provide a context to neuron activity. The ability to reliably identify these behaviors can allow scientists to correlate brain activity to behavioral functionality. Classifying these behaviors is a difficult task when performed manually but could theoretically be done through the use of machine learning methods. The MABe Challenge is a competition focused on evolving the understanding of such multi agent behavior prediction systems in order to remove the human component from behavior classifications. Specifically, in this challenge, a video of two mice has been encoded and mapped into a frame-by-frame coordinate system through an automated software pipeline titled MARS [8]. The resulting data set has been used to train neural networks to reliably predict behaviors. [The experiments that we performed intended to create and submit models that could reach the baseline of two of the tasks in the MABe Challenge.]

1. Introduction

Creating a reliable and robust behavioral classifier within the paradigm of the MABe challenge can be very insightful to the development of more abstract behavioral classifiers. Being able to understand how to best interpret an agent's actions can be very important in many research fields. The most notable is in the field of neurology; when looking at the brain activity of a mouse, the neural synapses and signals are only relevant with respect to the behavior exhibited by that mouse. Scientists have had to manually annotate and label each recorded video of the mice; which is a very slow and tedious task. This workflow creates an annotation bottleneck where the research is being limited by the rate at which they can accurately annotate their recorded raw data [1]. The ability to consistently and quickly identify the behavior of an agent will allow for neuro-scientists to extract more data and discover more insights about how the brain works.

The California Institute of Technology has provided frame-by-frame coordinate data of two mice by using a software called MARS [8]. Six trained experts have then annotated portions of the datasets with the correct behavioral classifications. With each annotation there is a annotator ID to be able to discern the discrepancies between annotators. This database of sequences was used in the investigation below to train supervised and unsupervised neural networks. In both the supervised and unsupervised portion of this project, the data labels have been annotated by the same individual thereby making the data less noisy and more consistent [1].

Reviewing previously developed models, the most successful attempts leveraged geometric and temporal metrics within a series of complex convolutional networks [9]. These metrics required preprocessing of the data where the input positional data was rendered into velocities and relative measurements between frames. These added complexities compelled the learning system to take into account specific temporal derivations as they are embedded into the network's calculations. The resulting performances assessed and correctly identified an average of 91% of all the behaviors when using all the available data. This model's architecture has directly inspired the work below.

The MABe challenge has driven the investigations of this report as it has organized that data and model goals into three parts; 1) supervised behavior prediction, 2) style transfer (between different annotators), and 3) new behavior learning (using low numbers of samples, or unsupervised learning). The provided datasets for tasks one and two, focus on 4 behaviors: attack, investigation, mount, and other. Task three is comprised of seven new and unique behaviors.

This report considers two of these tasks: Firstly, to develop a supervised learning model for task one with the goal of learning feature representations. Then, with this pre-trained model, perform new behavior learning on task three. Unfortunately, the server was down for task three which meant that the performance of the developed model could not be validated with objective metrics. This resulted in much more time invested in performing well on task one (supervised learning). However, unsupervised learning

[1] The entire dataset in the MABe challenge is annotated by all six annotators and is accessible to participants

methods were still explored for task three.

2. Approach

Behavior classifications are difficult and complex problems as the differences between two behaviors can be unnoticeable to even a trained scientists. These identifications become particularly difficult when labeling video data as it's very difficult for a human to determine when one behavior begins and ends on a frame-by-frame basis. Deep neural networks can solve this classification issue as they are more consistent and are unequivocally faster than any human. With a plethora of different available modules and tools, it can be difficult to design and train the right architecture for the right model. These networks will have to take into account the temporal nature of this behavior classification as a single behavior can best be identified through its context and not solely with respect to a single frame.

2.1. Losses, F1-score, and Regularization

A key aspect to the development of these models is the notion of success. Measuring success is a non-trivial task when considering a learning agent where over-fitting can be a serious consideration. The MABe competition provides training data but is insufficient large to use as validation. If the samples were to be split between training and testing sets, the resulting training would be inferior. Instead, the MABe challenge allows submissions to be uploaded and validated through hidden test sets. The returned evaluation metrics on behalf of these MABe tests were F1-Scores and Precision measures. These were the best objective methods of success to evaluate trained models. Between these submissions, three principle metrics were calculated to evaluate in-sample performances; loss, F1-score, and accuracy.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} = 2 \cdot \frac{\textit{precision} \times \textit{recall}}{\textit{precision+recall}} \quad (1)$$

The F1-score, shown in equation 1 is a balanced calculation between both precision and recall and it can be interpreted as a harmonic mean of both. This F1-score will be the primary metric to evaluate performance across all stand-alone models. The challenge's F1-score removes the majority class from the calculation and utilizes a macro average for the remaining classes. The loss was also considered at every epoch to verify model convergence. Low losses would indicate that the model has potentially saturated its learning capacity. High in-sample accuracy would also indicate that a model has thoroughly fulfilled its potential learning. A high accuracy model that has a high discrepancy between in-sample and validation scores will need to be re-designed to reduce over-fitting. Together, F1-scores, losses, and in-sample accuracy measures would allow model success to be evaluated and for over-fitting to be identified.

Regularization methods were also implemented to keep models from quickly over-fitting to the local training data. Early stopping was not possible because validation tests required time consuming predictions to be submitted and processed by the MABe competition servers. As mentioned, splitting the data into training and testing would be detrimental to the learning of the model due to the limited samples. Instead, dropout layers were implemented and data-augmentation mechanisms were designed to increase the quality and quantity of the available data. In turn, this would compel any trained model to learn more general feature representations and therefore be less likely to over-fit.

2.2. Supervised Models

Five particular models were considered for this application. Each of these used a series of frames to predict the behavior, not just the single target frame. The first was a basic LSTM model that used a fully connected layer to predict between the behavioral classifications. A second LSTM model was included that instead used a final CNN layer to output a prediction of the four classifications. A regular CNN model was also implemented that applied a series of convolutions to the input frames and returned the behavior predictions. A 1-dimensional ResNet [3] model was also developed as it is a notably successful network that is commonly used for image classification. Finally, a Transformer network was assembled to see if attention headers could provide utility to the prediction of these behaviors. An Ada-grad optimizer with a learning rate step-scheduler were used in combination with a focal loss [6] evaluation function. The adapted focal loss comes in the form of equations 2, 3.

$$\mathcal{L}_{FL}(p, y) = e^{-\gamma \cdot p_t} \times (e^p + 1)^{-\gamma} \times CE(p, y) \quad (2)$$

$$p_t = \begin{cases} p & \textit{if } y = 1 \\ 0 & \textit{otherwise} \end{cases} \quad (3)$$

This focal loss leverages the idea of hard classifications where some classifications are more difficult than others. In equation 2, the left-side factor, $e^{-\gamma \cdot p_t}$, only impacts the loss when there is a correct classification. When it correctly classifies, it exponentially scales the loss down the more confident it is with its prediction score p. Otherwise, the center term $(e^p + 1)^{-\gamma}$ will decrease with increased confidence in the score.

With the Ada-grad optimizer and this focal loss with a γ of 0.9999, the five models were independently tuned and compared to each other but only the CNN and Transformer models would be used for further investigation as discussed below in Section 3, Experiments. The initial developed models resulted in very mediocre results as the provided training data sets did not have balanced class distributions. This was a huge challenge with respect to both

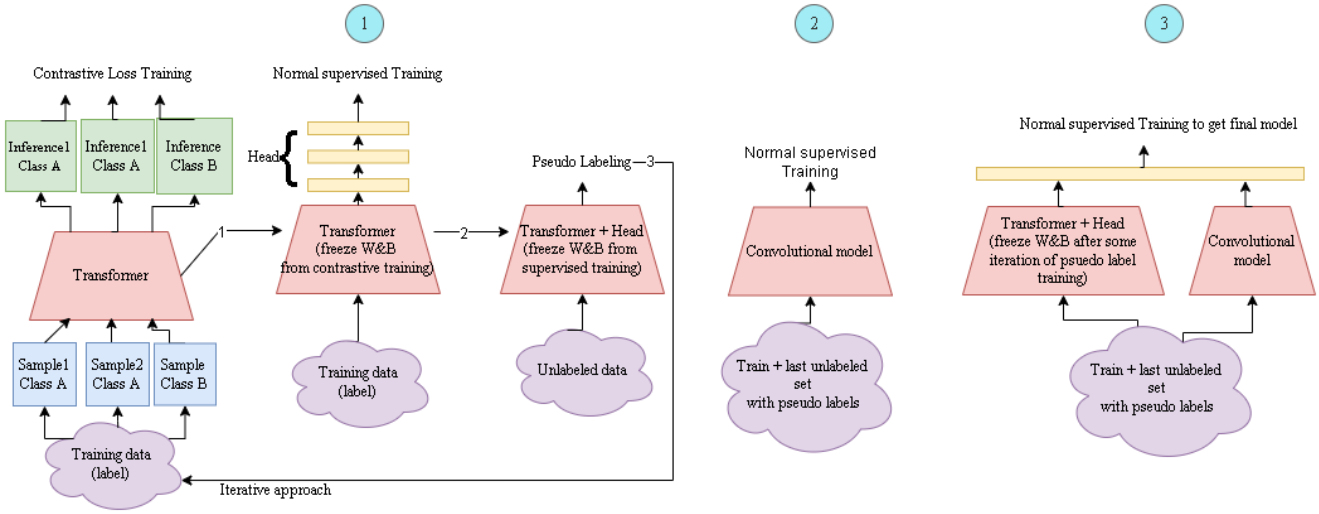

Figure 1. Semi-Supervised Model Training Diagram

the supervised and unsupervised learning tasks but was corrected through the use of a few methods. Firstly, weights were calculated through a class-balance loss factor in the form of equation 4, with a β value of 0.9999.

$$CB(n_y) = \frac{1-\beta}{1-\beta^{n_y}} \tag{4}$$

Additionally, a weighted sampler in combination with data augmentation mechanisms were implemented to help increase the impact of the under-represented and harder classes.

2.3. Semi-Supervised Models EM

With the development of high performance supervised models, a variety of new mechanisms could be explored. The architectures that were developed would be proved to be appropriate learners but, as mentioned above, were completely limited by the availability of labeled data. The MABe challenge also contained a larger data-set of unlabeled samples and thus could be used to generate pseudo-labels through knowledge distillation mechanisms as demonstrated by the state-of-the-art SimCLR approach [10].

A new loss mechanism would need to be implemented as traditional supervised loss functions rely on labeled targets. The triplet loss [2] function was used and came in the form of equation 6 where the output features of an anchor sample (A), would be compared to the outputs of a positive (P) and negative (N) sample.

$$\Delta\mathcal{P} = ||f(A) - f(P)||^2, \Delta\mathcal{N} = ||f(A) - f(N)||^2 \tag{5}$$

$$\mathcal{L}_{triplet}(A, P, N) = max(\Delta\mathcal{P} - \Delta\mathcal{N} + \alpha, 0) \tag{6}$$

Thus, firstly, contrastive loss training was implemented to pre-train a transformer with this triplet loss function. The available unlabeled training data would be used to generate an output of 64 feature classifications in order to extract samples for the triplet loss. Then, after attaching a FCN head and freezing the transformer's weights and biases, the head would be fine-tuned with regular supervised training via a class balanced focal loss criterion. This final network (the combined transformer and head) would then act as a teacher where it could process the unlabeled data and produce a larger set of pseudo labeled data. In combination with the regular labeled data, this expanded data-set could then be used to train superior student models.

Then, as demonstrated in the established methodology titled as "Self-Trained Noisy-Student" [7], this knowledge distillation process would be expanded to function as an iterative approach. After generating pseudo-labels from the teacher model, a student model would train on these pseudo-labels and be compared to the teacher model by testing on the original labeled data. If the resulting student model would provide superior results than the teacher, the teacher would be eliminated and the student would be promoted to be the new teacher and generate a new batch of pseudo-labels. This process would be iterated several times and would eventually result in producing the highest performance model.

Finally, the resulting model could be used to develop an ensemble model in combination with the successful CNN network as shown in figure 1.

2.4. Unsupervised Models JB

Another method for learning features is by performing unsupervised learning. This is done using unlabelled data via the triplet loss which is give in equation (6) from section 2.3. In this case it would be full unsupervised learning. The positive sample is a data augmentation of the anchor point and the negative sample is a random draw off a queue of past anchor-points. The feature representations learned

by this method can be then be leveraged to classify new behaviors by fine tuning using a small set of labelled data. Secondly, transfer learning from the task one model could be used to leverage feature representations that are known to perform well on task one. These features can then be leveraged to perform embeddings on task three data and through fine tuning perform classifications on new behavior. The goal is to understand if supervised labels of different behaviors (from task one) can be leveraged to learn new behaviors (from task three).

3. Experiments

In the litany of architectures that exist across the paradigm of deep learning, there is a set of models that can best perform in this situation. A properly sized network will always optimize with respect to its criterion as the loss will inevitably decrease to a limit. Loss is only as good as the precision and recall of the resulting optimized network. The F1 metric, defined in equation 1, balances both precision and recall and is the primary measure used by the MABe competition to evaluate and rank model submissions.

3.1. Data Augmentation

Data augmentations were imperative to the performance of the models. Due to the limited data, it was important to create a reliable set of training samples that could increase the robustness of the model. Enhancing these training samples resulted in a reduction of over-fitting as the augmented samples compelled the model to pick up on more general features and not over fit to specific features unique to the training data only. Several different augmentations were implemented as shown in figure 2. Within a 30 epoch

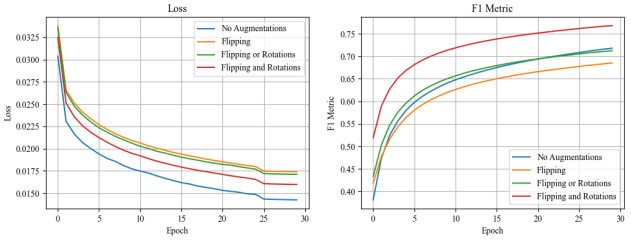

Figure 2. Model Augmentation Impact

limit, various training tests were run at different levels of augmentations. Augmentations had a direct impact on the losses as adding a simple flipping increased the losses by 15%. This is expected behavior considering that the unaugmented data is more homogeneous. Yet, as more augmentations were performed, the losses began to decrease again. When considering the F1-scores, it can be noted that although the losses increase, the actual performance of the model can improve. Yet, it must be noted that some augmentations may not actually increase the network's feature representations as F1-scores for the flipping augmentation decreased the actual performance with respect to the "No Augmentation" scores. But with a substantial enhancement to the training samples; the model actually improved its learning over the entire training. This shows that augmentations can be uniquely useful when applied correctly.

3.2. Supervised Learning Candidate Models

The unique issue with the supervised learning task is that the data is limited and it is impossible to accurately and proactively validate the model without sacrificing a good portion of the training data. This is because the test-set is unlabeled and is only evaluated when uploading the test-set classifications to the MABe submission server. It is therefore difficult to detect over-fitting before submitting the test-set results. The only consistent method to verify good model predictions is by submitting the test-set results at epoch intervals. With the many potential models, the only way to measure them was to use the same evaluation measure: the F1 metric. Locally, in-sample accuracy and F1-scores are measured to evaluate the candidacy of that model and then they are compared to the MABe evaluated test validation F1-scores. Models that have a high local accuracy and an in-sample F1-score that overfits the validation F1-score are likely to result in very bad performance. Not only is the model over-fitting but it has such high accuracy that it doesn't have much more to learn as it's already perfectly predicting the training classifications.

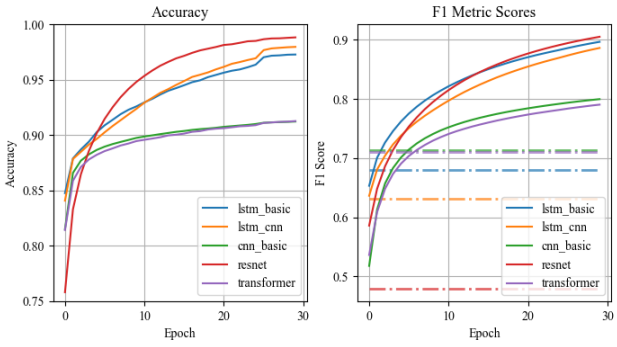

Figure 3. Model Comparison

Figure 3 shows the results of the first experiment where each candidate model was run for 30 epochs. An Ada-grad optimizer was used in accompaniment with a class-balanced focal loss criterion. As can be noted by these results, the resnet model quickly and effectively reaches an in-sample accuracy that approaches the 100% mark. Yet, when considering the F1 metric, and the dotted true objective F1-score evaluated by the MABe competition, the resnet is evidently undergoing extreme over-fitting. It's high accuracy is indicative that the network has picked up on features within the data but specifically only within the training set. This

Model	F1 Score	Precision	Runtime
LSTM Basic	0.680	0.719	03:23
LSTM CNN	0.631	0.653	03:38
CNN	0.714	0.781	00:28
Resnet	0.479	0.557	03:32
Transformer	0.710	0.813	00:46

Table 1. 30 Epoch Validation Scores

same flawed behavior is exhibited by both the LSTM models, leaving the CNN and Transformer models to be the only candidates for further investigation. For this reason, these two models are invalid networks to be used in the later experiments. Furthermore, as detailed in table 3.2, they are the slowest models to train. It could be possible to adjust the capacity and size of each model such to help it learn the data more appropriately, but considering these runtimes, it may not be worth the effort when the findings of figure 4 already show promising results from the CNN and Transformer models.

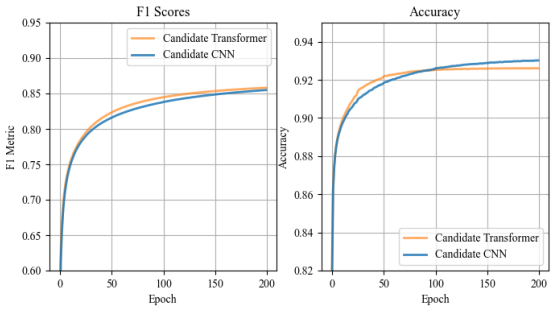

Figure 4. Candidate Transformer and CNN Comparison

The CNN model performs well in figure 3 but after more investigation, it fails to capture the full capacity of the environment. As can be understood from figure **??**, the CNN model continues learning from its training data and the in-sample F1-scores continue to increase. Yet, with intermittent validation tests, it is clear that the model was not improving on the actual test sets. The Transformer, similarly learns consistently but outperforms all other models throughout the learning process. Shown in figure **??**, similar to figure **??**, the Transformer reaches higher true F1-scores and shows more promising results. Similarly, it has issues overfitting as the in-samples scores are significantly higher than the validation results.

3.3. Semi-Supervised Learning DL

Due to the faster training of the Transformer model and highest F1-score achieved, we performed more experiments with it for implementing the semi-supervised approaches described in section 2.3. We trained multiple base models with Transformers with Contrastive Loss, mostly exploring different output sizes (12, 32, 64 and 128). We also tuned the dimension of the feed forward layer and the dimension of the key-value-query layer. In a second step, we trained Head Backend models with Focal Loss for fine-tuning, achieving even greater performance. For a complete list of hyper parameters, reference Appendix 1.

This enhanced model was used as a first teacher to generate pseudo-labels on the test set that later supported the iterative approach, described in Figure 1 diagram 1. We continued the process by training a new Transformer model based on the original labeled train set and the test set with pseudo labels. A similar approach was used to train the CNN model, and for both, we trained a Head Backend that achieved their best individual F1-score. Finally, we did an Ensemble model, as shown in diagram 2 of the same figure, training a final FCL for a few epochs, to obtain a F1-score that surpassed the baseline of the task 1 MABe Challenge by a few decimal points.

3.4. Unsupervised Learning

Unfortunately, as mentioned in section 1 we were unable to evaluate over test data for task three due to server issues on the AICrowd website. To compare relative feature representation in the unsupervised space and the potential clustering of each feature representation, two low dimensional embedding method were used; 1) TSNE plots and 2) Multi-Dimensional Scaling methods (MDS)[5]. This would allow for the visualization of the clustering of arbitrary behaviors and give insight into the effectiveness of learned features. MDS is used to show global structure that is present in the trained embeddings. While some local detail is lost, this is still useful to characterize seperation on a larger scale. Figure 5 demonstrates this global structure. Additionally, figure 5 demonstrates the differences in embeddings when training in a completely unsupervised manner, versus leveraging labels from an unrelated task. Both of these subplots have learned a representation, but because labels aren't known in the unsupervised case, the loss function is merely trying to learn a representation that is robust to the data augmentations. This could lead to more efficient

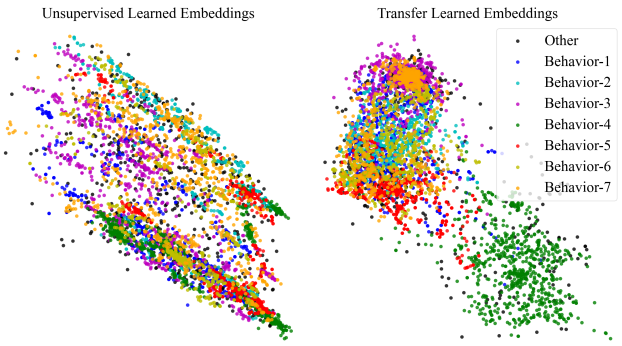

Figure 5. MDS Low Dimensional Embedding of Learned Features

Output Dim	Unsupervised Epochs	Ensemble Epochs	Head Epochs	F1 Score
12	200	100	None	0.769
64	300	200	None	0.779
64	300	200	CNN	0.783

Table 2. Ensemble Model Performance

representations that do not inherently learn discriminative features. However, in the Transfer Learning feature representation, even though the task is different behaviors, the model is attempting to separate different behaviors. This leads to more distinctive clusters as seen in Figure 5.

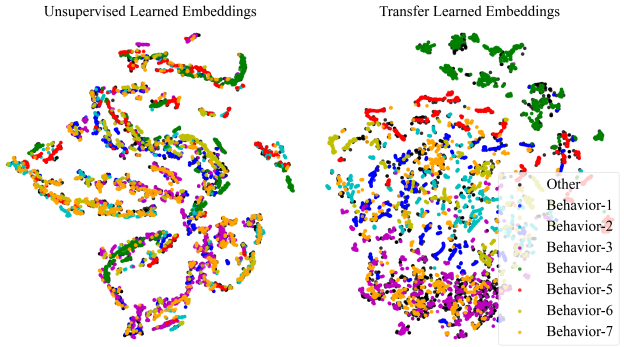

Figure 6. T-SNE Low Dimensional Embedding of Learned Features

Similarly, Figure 6 shows the same features embedded into a 2D space using T-SNE lower dimensional embedding. T-SNE characterizes more local structure. This gives more insight into what the feature embeddings are learning. It can be seen that the unsupervised features tend to keep the mouse path trajectories close to each other in path like structures. While, conversely, the transfer learned embeddings tend to separate out different groups. This could be why the transfer learned embeddings performed better than the unsupervised learning. During the last weekend, the AICrowd server came back online and we were able to evaluate both methods on the test set. The transfer learned method performed best with an F1 score of 0.228 and Precision of 0.188. The unsupervised feature learned method had an F1 score of 0.096 and a precision of 0.084.

4. Results

Two of the candidate models showed promising results as the CNN and Transformer provided reasonable F1 scores both with the in-sample training data and the validation tests from the MABe submission results. Both models exhibited over-fitting but were then improved through the data-augmentation techniques and with hyper-parameter tuning.

	CNN		Transformer	
Behavior	Precision	Recall	Precision	Recall
Attack	0.8642	0.7247	0.9033	0.7902
Investigation	0.8368	0.8910	0.8697	0.8973
Mount	0.8528	0.9107	0.8740	0.9290
Other	0.9616	0.9364	0.9618	0.9494

Table 3. CNN In-Sample Performance

4.1. Convolutional Neural Network

The final one-dimensional convolutional network performed extraordinarily well without the basic training set from task one. All behaviors were accurately classified to a minimum precision of 83% although it was not able to necessarily classify all the relevant classifications correctly as the recall of the attack behavior scored a low value of 72% as shown in table 4.2. This network trained for 200 epochs and consisted of four 1D convolutions of size 94 and a kernel size of 3 (with sequential ReLU operators). Dropout layers (with probability 0.5) followed by max-pool operations (of kernel size 2) were included after the first and third convolutions. The resulting output was passed through a 4 layer fully connected network with ReLU non-linearities to bring the dimensions down to an output of size 4. Each convolution allows the network to extract a particular feature as each pooling layer reduces the dimensional carrying only the dominant features. The dropouts are placed specifically before the maxpool layers to compel the right features to become sufficiently representational such that the model will avoid overfitting only to a specific set of features.

The intuition behind the power of this network is based on its ability to identify patterns in the high dimensional data and project it onto a meaningful lower dimensional space. In its maxpooling, it may be encoding the velocities and accelerations of the mouse positions as the max between two adjacent frames will differ depending on how quickly the coordinates change across each sequential frame. The high performance attempt performed by a previous institution [9] embedded velocities and accelerations directly into their models and this convolutional-maxpool network may be learning to do these independently.

4.2. Transformer Network

A transformer is a unique attention-based network that considers context into its predictions. The particular architecture implemented into this refined model uses a contextual embedding size of 192 and a feed-forward dimension block of 1024 to avoid a potential back-propagation bottleneck. The attention mechanisms work with key-value-query layers of dimensions 64. It is these attention layers that enable the network to make predictions solely with attention mechanisms [11]. The transformer model performs far better than the CNN network, scoring similarly or higher

Model	F1 Score	Precision
CNN	0.786	0.809
Transformer	0.791	0.810
Ensemble	0.803	0.817

Table 4. Final Model Validation Performance

across all behavior classifications.

It is the key-value-query layers that provide the high performance of this model as it is able to focus on the most important movements of the mouse. This mechanism forms a multiplicative interaction between the frames and coordinates that result in higher order relationships between each frame [4]. A convolution is fundamentally a complex linear operation (without the non-linearities), but the transformer is a polynomial operation as the input is multiplied by itself. This consequentially suppresses or magnifies features and representations within the input. Evidently, this provides great results nearly beating the benchmark value provided by the MABe challenge.

4.3. Ensemble Semi-Supervised Model

The Transformer and CNN were the fastest to train and reached the highest scores in the MABe validation tests and were then selected to pursue the methodologies discussed in section 2.3, *Semi-Supervised Models*. Using the Transformer trained with supervised data, batches of unsupervised data were interpreted into larger sets of pseudo-labeled data. With these larger sets, new CNN and Transformer models were pretrained through contrastive methods and refined with an appended FCN head and finally ensembled, as seen in Figure 1, diagram 2. Their weights were then frozen and training a final FCN head on supervised data achieved the highest F1-score from the MABe Challenge achieving our goal of reaching and surpassing the baseline. For reference see Table 4.3

4.4. Unsupervised Models

Triplet losses were an extremely time consuming process (approximately 3.5 hours per epoch) due to the large amounts of unlabelled data. The first implementation was to develop enhancements that would decrease the learning time. Firstly, the augmentations that the supervised learners were using were bypassed for all samples when using the triplet loss. To get a positive sample, the nearest neighbors were directly sampled (n time steps from the anchor sample) to obtain a matching data point. Then, for a negative sample, a queue of past anchor points was sampled to find a contrasting data instance. This sped up the implementation to less than 2 hours per epoch. However, this came at the cost of having a loss close to zero during mini-batch optimization updates due to anchor points and positive samples being too near. Therefore, the method reconciled by only using augmentations for positive samples only.

Additionally, there was not a way to evaluate the success of the unsupervised learning for task three. Therefore, development time was spent to create low dimensional embedding plots for visual inspection. During the last weekend of the project, the server became available for evaluation. This allowed direct evaluation of transfer learning vs. unsupervised learning for task three as discussed in section 3.4.

5. Extended Approach

talk about other things that didn't work

SimCLR approach [10], we were able to achieve only achieving 0.56 F1 score for task 1

Already mentioned but once again, for the Triple Loss in task 3 we were only able to train for a few epochs due to that training time was taking around two hours and we didn't have constant access to GPUs for more than 10 hours continuously. We only try for 15 epochs due to the previous, we think that if we try for more epochs maybe around 100, we would achieve better results for task 3.

References

[1] Jordana Cepelewicz. To decode the brain, scientists automate the study of behavior. *Quantam Magazine*. 1

[2] Nir Ailon Elad Hoffer. Deep metric learning using triplet network. *ICLR 2015*. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[4] Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. 2020. 7

[5] J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964. 5

[6] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 2

[7] Xie Q. and et al. Self-training with noisy student improves imagenet classification. abs/1911.04252, 2020. 3

[8] Cristina Segalin, Jalani Williams, Tomomi Karigo, May Hui, Moriel Zelikowsky, Jennifer J. Sun, Pietro Perona, David J. Anderson, and Ann Kennedy. The mouse action recognition system (mars): a software pipeline for automated analysis of social behaviors in mice. *bioRxiv*, 2020. 1

[9] Jennifer J. Sun, Tomomi Karigo, Dipam Chakraborty, Sharada P. Mohanty, David J. Anderson, Pietro Perona, Yisong Yue, and Ann Kennedy. The multi-agent behavior dataset: Mouse dyadic social interactions. *CoRR*, abs/2104.02710, 2021. 1, 6

[10] Chen T. and et al. Big self-supervised models are strong semi-supervised learners. abs/2006.10029, 2020. 3, 7

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 6

Student Name	Contributed Aspects	Details
Emilio Moyers	Data Loaders, Augmentations, Resnet model, Noisy Student, Ensemble, and Unsupervised pre-training.	• Implemented three Data Loaders used in task 1 and task 3. • Implemented three data augmentations used by task 1 and task 3. • Implement 1-dimensional ResNet and run experiments using it for task 1. • Implement noisy student pipeline to create pseudo-labels, run experiments using the pipeline with the best models (Transformer and CNN). • Implement ensemble to combine Transformer and CNN. • Implement unsupervised pre-training pipeline for task 1 and task 3 using SimCLR as reference.
Diana Lomelin	Augmentation, Train-test split, Test Data Loader, F1-score, Transformer Models, Partition of test set, Parametric Analysis, Ensemble	• Implemented basic data augmentations used by task 1 and task 3. • Implemented a Data Loader for an augmented testing set used in task 1. • Implemented the F1-score metric for the in-sample analysis. • Implemented 1-layer Transformer and run experiments using it for task 1. • Implemented a multi-head, multi-layer Transformer and run experiments using it for task 1. • Helped implement ensemble to combine Transformer and CNN. • Implemented partition of test set for unsupervised learning in task 1. • Performed Parametric Analysis on the Transformer, Transformer with Head Backend, and Ensemble models.

Table 5. Contributions of team members.

Student Name	Contributed Aspects	Details
Jonathan Barton	Triplet Loss Training, CNN Model, CNN-LSTM Model, Head Backend Fine tuning model, Task 3 Submission Code, MDS and TSNE Embeddings, Task 3 Transfer Learning	• Implemented the Triplet loss for training. This includes implementing supervised use of triplet loss as well as unsupervised. Additionally, implemented a queue for storing future negative samples to speed up triplet loss for Task 3. • Implemented three models for testing: 1D CNN Model, CNN-LSTM Model, and a linear model for supervised fine tuning "Head Backend". • Implemented two additional data augmentation methods. • Implemented the supporting code for validating and submitting Task 3 results. • Created MDS and TSNE modules to create visualization charts for Task 1 and Task 3 visualization. • Used model trained on Task 1 to create feature embeddings on Task 3 data and train a supervised fine tuned model for Task 3 new behavior learning.
Andres Menendez	Task 1 Modules, Parametric Analysis, Report Figures and Discussion	• Implemented modules for performing Focal Loss including class-balance and focal loss. • Implemented all of the model-submission code as well as assisted with model development. • Trained all models with different sizes and hyperparameters. • Collected data and performed hyperparameter adjustment. Additionally, compared models with respect to each other and data augmentations.

Table 6. Contributions of team members – Continued from previous page.