Of Mice and Men : CS 7643

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

Abstract

The brain is 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 allows scientists to correlate brain activity to behavioral functionality. Classifying these behaviors is difficult 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 these 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.

1. Introduction

Creating a reliable and robust behavioral classifier within the paradigm of the MABe Challenge can be 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 neurology; when looking at the brain activity of a mouse, the neural synapses are only relevant with respect to the exhibited behavior of the animal. Scientists have to manually annotate each video of the mice; which is slow and tedious. This creates an annotation bottleneck where the research is limited by the rate at which the recorded raw data can be labeled [1]. The ability to consistently and quickly identify the behavior of an agent will allow neuro-scientists to extract more data and discover more insights about how the brain works.

The California Institute of Technology provided frame-by-frame coordinate data of two mice by using a software called MARS [8]. Six experts have then annotated portions of the data-sets with the correct behavioral classifications. Each annotation has an annotator ID to discern the discrepancies between annotators. Only a portion of this database was used in the investigation to train supervised and unsupervised neural networks, specifically, all the sequences used were annotated by the same expert.

Previous successful models leveraged geometric and temporal metrics within a series of convolutional networks [10]. These models used preprocessed 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 model correctly identified an average of 91% of the behaviors when using all the data. This model's architecture has directly inspired the work below.

The MABe Challenge has driven the motivations of this report as it provided data for three tasks; 1) supervised behavior prediction, 2) style transfer (between different annotators), and 3) new behavior learning (using low samples counts, 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 challenge was inactive and acted only as a motivation and as a source of model validation testing. Their platform provided the data sets, starter code and objective evaluation methods. See Appendix A. Although, the starter code was provided in TensorFlow, all the developed models used a newly created Pytorch framework.

This report considers two of these tasks: Firstly, to develop a supervised learning model for task one with the goal of learning feature representations and reach the baseline F1-score of 0.791 and then perform new behavior learning on task three. Unfortunately, the server was down for task three and the model could not be validated with objective metrics.

2. Approach

The following sections describe the approach taken to achieving the baseline performance from the MABe Challenge. The development was split into two main categories. First, the loss functions were improved to learn discriminative features on the dataset and to classify under-represented

classes well. The methods for this include implementing Focal Loss, Triplet Loss, Weighted Sampler, and data augmentation methods. Second, model development resulted in a variety of models to select from for final evaluation. Third, extending the supervised dataset through teacher student data distillation was performed to squeeze the most performance out of the existing data.

2.1. Losses, F1-score, and Regularization

The MABe competition provides training data but is insufficiently large to use as validation. The MABe platform 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 to evaluate the success of 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 their harmonic mean. 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 also indicates that a model can not improve on the training data-set. A high accuracy model that has a high discrepancy between in-sample and validation scores will need to be re-designed to reduce overfitting. Together, F1-scores, losses, and in-sample accuracy measures would allow model success to be evaluated and 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 due to the MABe platform's limited number of submissions and slow validation time. Additionally, splitting the data into training and testing would be detrimental to the learning of the model due to the limited samples. Instead, regularization methods such as weight decay and dropout along with data augmentation techniques were implemented to prevent over fitting.

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 two models were LSTMs [9] that used either a fully connected layer to predict classifications or a CNN to learn spatial features that were then fed into the LSTM . 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 commonly used model for image classification. Finally, a Transformer network was assembled to see if attention headers could provide utility to the prediction of these behaviors. Since the feature space is continuous for this problem, unlike a language-related problem with a finite number of tokens to embed, an alternative embedding of the 28 features (7 body parts, by 2 coordinates, by 2 mice) was used, passing them through a linear layer to represent them in the size of embeddings that were desired. They were then then added to the positional embeddings in a regular fashion. This alternative method seemed to have captured features better that the built-in methods for Transformers that Pytorch provides. An Ada-grad optimizer with a learning rate step-scheduler was 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. This effectively dampens the impact that easy-classifications have on the loss while weighing the loss from harder classifications and miss-classifications higher.

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 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.

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

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

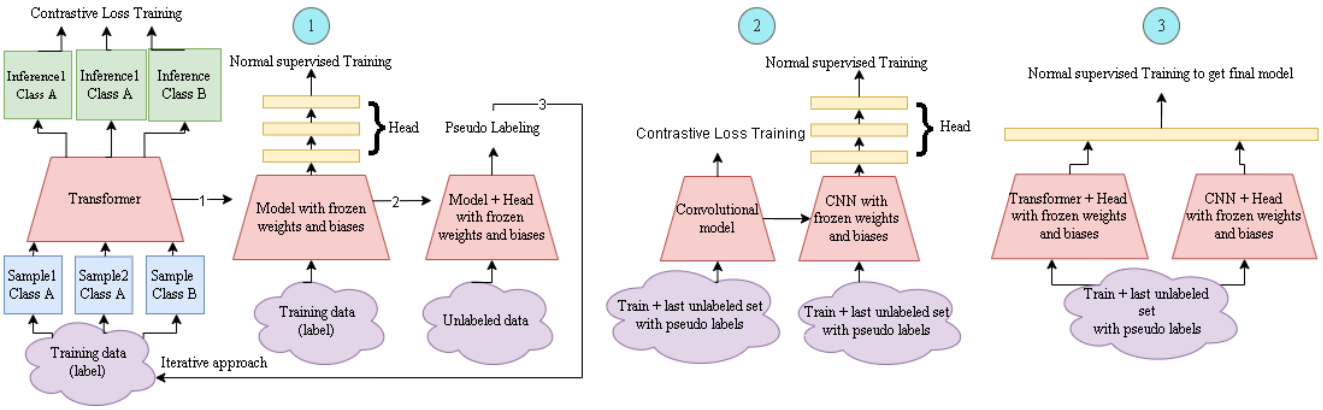

Figure 1. Semi-Supervised Model Training Diagram

2.3. Semi-Supervised Models

To improve past the capabilities of solely supervised learning models, semi-supervised learning techniques were implemented for this challenge. The MABe challenge contained a large 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 [11].

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.

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

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

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. Learning New Behavior

Leveraging the unlabelled data and triplet loss which is given in equation (6) from section 2.3, *Semi-Supervised Models*, the models were able to learn feature embeddings for new behavior learning (MABe Challenge task three). To train triplet loss on the unlabelled dataset, With respect to the triplet loss, a positive sample is an augmented variation of the anchor point and a negative sample is a random draw from 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 small sets of labelled data. Secondly, the feature representations learned in task one could 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 used to learn new behaviors (from task three).

3. Experiments

The following sub-sections details the experiments that were performed to select the method of training as well as the model architecture for the task. Additionally, the figures show qualitative results that drove the decisions which culminated in above baseline performance.

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 limit,

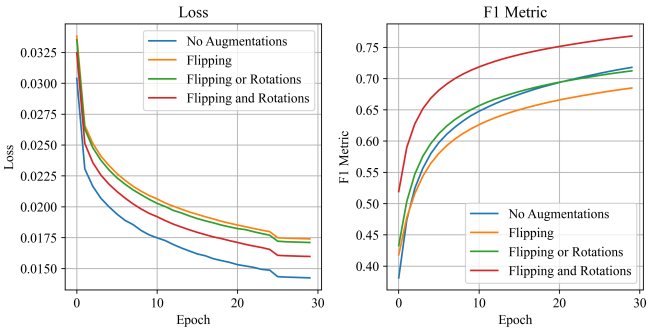

Figure 2. Model Augmentation Impact

various training tests were run at different levels of augmentations. They had a direct impact on the losses as adding a simple flipping increased the losses by 15%. This is expected behavior considering that the un-augmented data is more homogeneous and it is therefore easier for the criterion to perform gradient descent. Yet, with additional augmentations, the losses began to decrease again as the additional augmentations compel the network to learn more appropriate features and therefore learn more efficiently. This is most apparent when considering the F1-scores where sufficient enhancements result in improvements in learning and in the entire training. This shows that augmentations can be uniquely useful when applied correctly.

3.2. Supervised Learning Candidate Models

The primary disadvantage to relying on supervised learning, 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 platform validation scores. Models with high local accuracy and an in-sample F1-score that overfits the validation 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 shows the results of the first experiment where each candidate model was run for 30 epochs with an Ada-grad optimizer and class-balanced

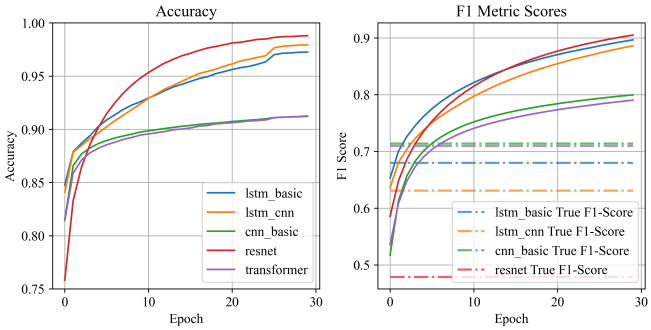

Figure 3. Model Comparison

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

focal loss criterion. The ResNet model quickly reaches an in-sample accuracy of nearly 100%. but dramatically overfits when as the dotted true objective F1-score measures at 0.479. This same flawed behavior is exhibited by both the LSTM models. Additionally, when considering the runtimes in table 1, only the CNN and transformer would be the most suitable candidates for further investigations.

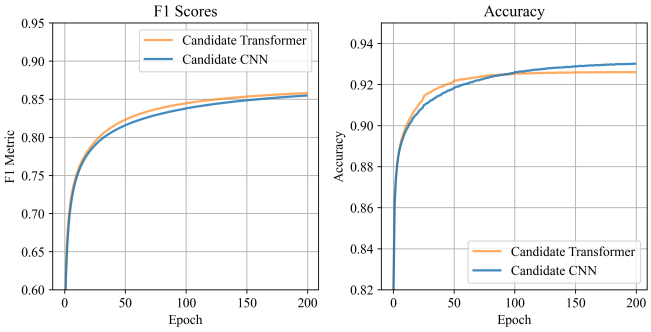

Figure 4. Candidate Transformer and CNN Comparison

Although the CNN and transformer performed well within the 30 epoch range, further tests were run to verify their validity in a larger context of 200 epochs with the same hyper-parameters. Shown in figure 4, both the CNN and Transformer perform very well and demonstrate equal potential. Instead of choosing a single final model, both will be chosen together to act in union as an ensembled model.

3.3. Semi-Supervised Learning

The successful CNN and transformer architectures don't just provide the potential for an ensemble learner but inspire the experimentation for semi-supervised approaches described in section 2.3. Multiple tuned transformers were trained with a contrastive loss method through the exploration of different output sizes (12, 32, 64 and 128). The

feed-forward layer and the dimensions of the key-value-query layers were also fine-tuned to provide the best possible results. In a second step, Head Backend models were trained with focal loss fine-tuning, and thereby resulting in even greater performances.

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. The process continued 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, as the Head Backend was trained to achieve the best individual F1-score (step 2). Finally, an ensemble model was implemented as shown in step 3 of Figure 1, training a final FCN for only 10 epochs, to obtain an F1-score that surpassed the baseline on task 1 of the MABe Challenge by a few decimal points. For a complete list of hyper-parameters, reference Appendix B.

3.4. Unsupervised Learning

Unfortunately, as mentioned in section 1 the models were unable to evaluated 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 methods were used; 1) T-SNE plots[12] 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 feature space. While some local detail is lost, this is still useful to characterize separation on a larger scale. Figure 5 displays 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 features that are more representational rather than discriminative. However, in the Transfer Learning feature representation, even though the features were trained on task one behaviors, the features learned may be more discriminative. wich leads to more distinctive clusters as seen in Figure 5.

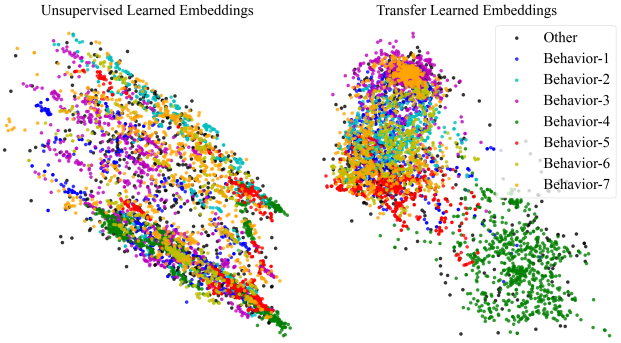

Figure 5. MDS Low Dimensional Embedding of Learned Features

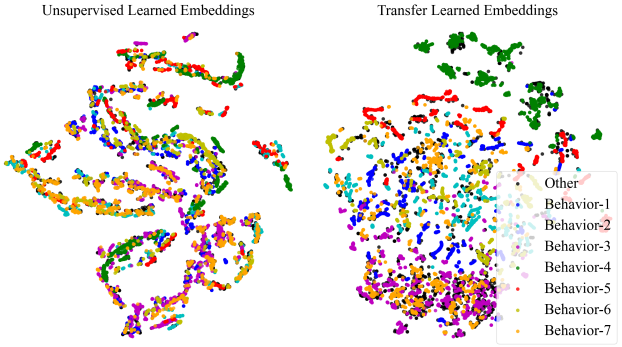

Figure 6. T-SNE Low Dimensional Embedding of Learned Features

Similarly, figure 6 shows the same features embedded into a 2D space using T-SNE's lower dimensional embeddings. T-SNE characterizes more local structures rather than global. This gives additional insights into the structure of learned features. The unsupervised features tend to keep path trajectories like structures while the transfer learned embeddings separate out different sub clusters. This could be why the transfer-learned features performed better than the unsupervised learning. During the last weekend of this effort, the AICrowd server became available and both methods were evaluated 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

4.1. Convolutional Neural Network

After confirming the validity of the CNN in section 3.2, more efforts were put into fine-tuning the CNN parameters. The best model achieved a classification precision of 83% although it was not able classify all the relevant classifications correctly as the recall of the attack behavior scored a low value of 72% (shown in table 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.

	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 2. Model In-Sample Performance

The intuition behind the power of this network is based on its ability to identify spatial patterns that are useful across all sequences. With large 1D convolutions, more relationships across the temporal plane can be identified and used to accurately detect features. The max-pooling layers help reduce dimensionality and also take in some temporal representations as the max of sequential frames differs depending on how fast the features change across frames.

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 [13]. The transformer model performs far better than the CNN network, scoring similarly or higher 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 3.

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

Table 3. Final Model Validation Performance

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 being too similar to the positive samples. 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. Future Work

The semi-supervised approach in combination with the ensembled networks proved to be a very nuanced and exciting demonstration of the power of deep learning. It was a novel approach as far as previous documentation shows and it successfully passed the MABe baseline score by a healthy margin. Section 3.4 demonstrated great potential but unfortunately it could not be validated in time. Future efforts could focused on fine-tuning the final ensembled model and being able to accurately measure the unsupervised portion of the project.

References

[1] Jordana Cepelewicz. To decode the brain, scientists automate the study of behavior. *Quantam Magazine*, 2019. 1

[2] Nir Ailon Elad Hoffer. Deep metric learning using triplet network. *ICLR 2015*, 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. *ICLR 2020 Conference Blind Submission*, 2020. 6

[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. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR*, abs/1909.09586, 2019. 2

[10] 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

[11] Chen T. and et al. Big self-supervised models are strong semi-supervised learners. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, abs/2006.10029, 2020. 3

[12] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008. 5

[13] 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

Appendices

A. Multi-Agent Behavior Challenge

The following website contains the official MABe Challenge description, data sets, starter code and submissions page that we used to complete this project.
MABe Challenge

B. Hyper-parameters

See Table 6.

C. Hyper-parameters Unsupervised

See Table 7.

D. Project Code Repository

The Github repository for our final project can be found at:
Github Team Of Mice and Men

E. Project Trained Models

Some of the models trained during the proect can be found at:
Models Trained

F. Parametric Analysis Spreadsheet

A spreadsheet with the Parametric Analysis can be found at:
Github Team Of Mice and Men - Documents

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 4. 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 loss evaluation 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. • Ran experiments, and collected data • Created diagrams and figures for results and analysis • Wrote outline, introduction section and proofed final version of report

Table 5. Contributions of team members – Continued from previous page.

Modules	model	LSTM_basic	LSTM_CNN	CNN_basic	Resnet1D	Transformer_basic
	criterion	CBFL*	CBFL*	CBFL*	CBFL*	CBFL*
	optimizer	Adagrad	Adagrad	Adagrad	Adagrad	Adagrad
	scheduler	None	None	None	None	Step (.9 at 20)
Model Parameters	learning_rate	1.00E-04	1.00E-04	1.00E-04	1.00E-04	0.001
	momentum	0.5	0.5	0.5	0.5	0.5
	weight_decay	0.00001	0.00001	0.00001	0.00001	0.00001
	num_epochs	20	20	20	20	20
	batch_size	512	512	512	512	512
Linear Model	layers	4				
	hidden_size	512				
CNN Model	cnn_width	-	14	14		14
	cnn_layers	-	4	4		4
	cnn_kernel_size	-	3	3		3
Transformer Model	output_size	-	-	-	-	4
	hidden_dim					192
	num_heads	-	-	-	-	2
	dim_ff	-	-	-	-	1024
	dim_kvq	-	-	-	-	64
Data Loader	past_frames	10	10	10	10	10
	future_frames	10	10	10	10	10
Training / Validation	test_fromsplit	TRUE	TRUE	TRUE	TRUE	TRUE
	split_size	0.2	0.2	0.2	0.2	0.2
Augmentation	weighted_sampler	FALSE	FALSE	FALSE	FALSE	FALSE
Triplet Loss	triple_loss	FALSE	FALSE	FALSE	FALSE	FALSE
	margin	1	1	1	1	1
	p_norm	2	2	2	2	2
CBFL *	beta	0.99	0.99	0.99	0.99	0.99
	gamma	5	5	5	5	5
Prediction	single_prediction	TRUE	TRUE	TRUE	TRUE	TRUE

Table 6. Initial Models Hyper-parameters for Supervised Learning.
* Class Balance Focal Loss

Hyper-	CNN		Transformer	
parameter	Base	Head	Base	Head
Learning rate	0.001	0.001	0.001	0.001
Momentum	0.5	0.5	0.5	0.5
Weight decay	0.00001	0.00001	0.00001	0.00001
Number epochs	200	100	300	100
Batch size	512	512	512	512
Number classes	4	4	4	4
Feature size	28	28	28	28
Output size	64	64	64	64
Past frames	10	10	10	10
Future frames	10	10	10	10
Sequence length	21	21	21	21
CNN width	94	-	-	-
CNN layers	4	-	-	-
CNN kernel size	3	-	-	-
Transformer hidden dim	-	-	192	-
Transformer number heads	-	-	2	-
Transformer dimension feed-forward	-	-	1024	-
Transformer dimension kvq	-	-	64	-
CBFL gamma	-	5	-	5
CBFL beta	-	.9999	-	.9999
Triple loss	True	False	True	False
Margin	2.0	-	2.0	-
P-norm	2	-	2	-

Table 7. CNN and Transformer Hyper-parameters for Unsupervised Learning