

Early Classifying Multimodal Sequences

Alexander Cao
Northwestern University
Evanston, Illinois, USA
a-cao@u.northwestern.edu

Jean Utke
Allstate Insurance Company
Northbrook, Illinois, USA
jutke@allstate.com

Diego Klabjan
Northwestern University
Evanston, Illinois, USA
d-klabjan@northwestern.edu

ABSTRACT

Often pieces of information are received sequentially over time. When did one collect enough such pieces to classify? Trading wait time for decision certainty leads to early classification problems that have recently gained attention as a means of adapting classification to more dynamic environments. However, so far results have been limited to unimodal sequences. In this pilot study, we expand into early classifying multimodal sequences by combining existing methods. We show our new method yields experimental AUC advantages of up to 8.7%.

KEYWORDS

early classification, sequence classification, multimodal sequences

1 INTRODUCTION

Early classifying multimodal sequences is ubiquitous in our lives. Scanning through movies on any streaming platform, the trailer begins to play. Based on seconds of video and audio, we make a quick decision of whether or not to watch it. A more serious example is a physician diagnosing a patient. From imaging, lab tests, and other results arriving at different times, the doctor is attempting to diagnose as accurately as possible (to start the correct treatment) as quickly as possible (to begin that treatment sooner). With such applications, it is important to adapt early classification methods to multimodal sequences as they exhibit their own challenges.

Specifically, early classification manifests in the following problem setup: At each time step we receive new information in a sequence. From here, we must decide if we have enough information to stop and predict the class with sufficient accuracy, or if we should continue waiting for additional information in later time steps to improve prediction accuracy. Balancing this dual objective of classifying as soon as possible, as accurately as possible is the core problem.

To solve this, we combine an OmniNet-like [14] transformer neural network with Classifier-Induced Stopping (CIS) [3]. Transformers [19] represent the state-of-the-art neural network for sequence-based tasks and OmniNet further advances them by explicitly modeling spatial-temporal interaction, making its architecture well-suited for our early classification of multimodal sequences. The recent CIS provides an efficient method to learn both a policy for deciding between stopping and waiting and a classifier. It works by finding the optimal stopping time based from its own classifications each time step.

Our contributions are two-fold. First, this is a pilot study in early classification of multimodal sequences. To our knowledge, this is first time early classifiers have been applied to sequences composed of different modalities such as images, text, and structured

categorical data. Second, we demonstrate that spatial-temporal transformers in combination with CIS is a potent model for early classifying multimodal sequences. Experiments show our method holistically outperforms a similar benchmark early classifier with the same neural network body.

This paper is structured as follows. In §2, we review related work in multimodal neural networks and early classifiers. In addition, we introduce relevant notation and detail the benchmark method. In §3 and §4, we thoroughly lay out OmniNet’s spatial-temporal transformer and CIS, respectively. All experimental details and results are explained in §5. Finally, we conclude in §6.

2 RELATED WORK

2.1 Neural Networks for Multimodal Sequences

There is a large body of work adapting LSTMs [9] to digest multimodal sequences. These methods can be distinguished by the stage of fusing of the different modalities. [17] concatenates features from different modalities and feeds this larger input into an LSTM. [1, 6] reserve separate LSTMs for each modality and then fuse the outputs. Finally, as a means of fusing modalities within a LSTM, [12, 15] have separate LSTMs for each modality but they share common weights. They argue this allows the model to jointly learn correlations across modalities.

More recently, however, transformers have been the answer to multimodal sequence-based tasks [23]. [24] captions images with a transformer composed of an image encoder to self-attend visual features and a caption decoder to generate the captions from those visual features. [18] introduces cross-modal transformers, which learn the attention between each pair of modalities. The downside being that as the number of modalities increases, the number of cross-modal transformers and model size necessarily increases too. OmniNet circumvents this issue by arranging elements of a multimodal sequence into a temporal and spatial cache and then feeds both into a spatial-temporal transformer. This mechanism allows temporal features to attend over the spatial features and implicitly learn a shared representation across multiple modalities. We present the full details of this spatial-temporal transformer in §3.

2.2 Early Classification

Initial works in early classification formulate the problem in terms of standard reinforcement learning. Specifically, [7, 11] use REINFORCE [22], a standard exploration-exploitation policy gradient method to learn their early classifiers. [3] uses PPO [16], another policy gradient method, to the same effect. These methods rely on trial and error and lack the ability to ‘look forward’ to see that waiting longer for more elements would have been beneficial. In fact, [3] experimentally demonstrates that PPO performs considerably worse than such forward-looking methods like Length

Adaptive Recurrent Model (LARM) [10] and CIS. Accordingly, we only benchmark CIS against LARM. Again, all previous work mentioned here is with unimodal sequences. Before summarizing LARM, in the next subsection we mathematically formulate early classification and establish notation.

2.3 Problem Setup Notation

The set of training data \mathcal{X} comprises samples $x^{(i)}$ and one-hot encoded labels $y^{(i)} \in \{0, 1\}^C$, where $C > 1$ is the number of classes and

$$x^{(i)} = \left((x_1^{(i)}, m_1^{(i)}), (x_2^{(i)}, m_2^{(i)}), \dots, (x_{T_{\text{end}}}^{(i)}, m_{T_{\text{end}}}^{(i)}) \right)$$

are sequences of elements $x_t^{(i)}$ of modality $m_t^{(i)}$. For a sample i at time t , its state is given by

$$s_t^{(i)} = \left((x_1^{(i)}, m_1^{(i)}), (x_2^{(i)}, m_2^{(i)}), \dots, (x_t^{(i)}, m_t^{(i)}) \right).$$

Classifier neural network f parameterized by α takes s_t as input¹ and outputs predicted class distribution vector $\hat{y}_\alpha(\cdot|s_t)$. Policy neural network g parameterized by β takes s_t as input and outputs policy distribution vector $\pi_\beta(\cdot|s_t)$ over two actions ('wait' and 'stop and classify now').

$$\hat{y}_\alpha(\cdot|s_t) = f_\alpha(s_t)$$

$$\pi_\beta(\cdot|s_t) = g_\beta(s_t)$$

At each time step t , we take an action a_t according to policy $\pi_\beta(\cdot|s_t)$. This action can be selected stochastically via sampling or deterministically by choosing the argmax action. We keep waiting another time step and receiving new elements (x_{t+1}, m_{t+1}) until we decide to stop. Once we decide to stop and classify, we make a classification according to $\hat{y}_\alpha(\cdot|s_t)$.

To learn classifying as accurately as possible, as quickly as possible, we implement the following reward function

$$R_t^\alpha(s_t, a_t) = \begin{cases} -\mu & \text{if } a_t = \text{'wait'} \\ -\mu - \text{CE}(y, \hat{y}_\alpha(\cdot|s_t)) & \text{if } a_t = \text{'stop'} \text{ or } t = T_{\text{end}} \end{cases}$$

where μ is a time penalty parameter and CE is cross-entropy. Each time step incurs a constant time penalty of $-\mu$. We denote the time the model stops and classifies as time $T \leq T_{\text{end}}$. Early classifying can then be formulated in terms of the following optimization problem

$$\max_{\alpha, \beta} \mathbb{E}_{\mathcal{X}} \sum_t R_t^\alpha(s_t, a_t(\beta)). \quad (1)$$

Maximizing this cumulative reward means classifying as accurately as possible (so that cross entropy is low), as quickly as possible (so that the sum of time penalties is low). The time penalty parameter μ captures how much waiting another time step is penalized. As μ grows, we may sacrifice more accuracy for earlier classifications, and vice-versa.

2.4 LARM

LARM [10] learns to early classify in a probabilistic manner. If A_T is the decision sequence where the policy decided to stop and classify at time T , then this decision sequence is uniquely defined by the sequence of actions

$$A_T = (a_1 = \text{'wait'}, \dots, a_{T-1} = \text{'wait'}, a_T = \text{'stop and classify'}). .$$

We can also explicitly calculate the probability of decision sequence A_T from policies $\pi_\beta(\cdot|s_t)$ as

$$\mathbb{P}(A_T|s_T) = \prod_{t=1}^T \pi_\beta(a_t|s_t).$$

With these stopping time probabilities, LARM seeks to maximize an expected cumulative reward based on (1) to learn its early classifier.

$$\min_{\alpha, \beta} \mathbb{E}_{\mathcal{X}} \left[\text{CE} \left(y, \sum_{T=1}^{T_{\text{end}}} \hat{y}_\alpha(\cdot|s_T) \mathbb{P}(A_T|s_T) \right) + \mu \sum_{T=1}^{T_{\text{end}}} T \cdot \mathbb{P}(A_T|s_T) \right]$$

The first term in this loss is a micro-averaged cross-entropy and the second term is the expected stopping time penalty. Again, for both terms the expectation is taken with respect to the stopping time T probability.

We see that if $\pi_\beta(a_t = \text{'wait'}|s_t)$ are small then $\mathbb{P}(A_T|s_T)$ may decrease to 0 rapidly. This is tantamount to not 'waiting' far enough into the sequence to gain valuable information. To prevent this, LARM sets the factors $\pi_\beta(a_t = \text{'wait'}|s_t)$ to 1 with probability ρ during training. This ensures the model will wait for more elements in the sequence initially. Again, we emphasize LARM is a capable of looking forward and learning when waiting will be beneficial. During inference, LARM follows a stochastic policy rollout (samples action $a_t \sim \pi_\beta(\cdot|s_t)$) but deterministically classifies.

3 SPATIAL-TEMPORAL TRANSFORMER

We follow OmniNet's [14] structure of first funneling elements of the multimodal sequence through their respective peripherals and then inserting those outputs in the temporal and spatial caches of the transformer segment. This section outlines this process, terminating with the policy and classifier decisions.

3.1 Peripherals

Before the state s_t reaches the transformer block of the neural network model, modality-specific peripheral functions are applied to each element in the sequence. For instance, an image peripheral is applied to image elements and a text peripheral is applied to text elements. If there are multiple sources of text elements, we can have a separate peripheral for each. The purpose of each modality (or source's) peripheral is two-fold: First, peripherals extract relevant features. Second, peripherals project each element to a common dimension size d_{model} , a necessity for a unified transformer. Consider an image of shape $(hw, 3)$ where h, w are the height and width of the image in pixels and 3 refers to the RGB channels. The image peripheral will project this image to dimension $(h'w', d_{\text{model}})$ where $h', w' > 1$ are reduced, downsampled height and width subpixels. For text, there is no spatial dimension so we write their shape as $(1, n)$ where n is the number of words or tokens. Similarly, the text peripheral will project this text to dimension $(1, d_{\text{model}})$.

¹We only explicitly write superscript samples $x^{(i)}, s^{(i)}$ when needed to distinguish.

Figure 1 depicts the overall structure of our neural network architecture with a didactic peripheral flow example. Say the first element x_1 of state s_t has modality $m_1 = \text{image}$. Note, images are the only spatial modality in our case. The image peripheral is applied and the resulting output is appended to the spatial cache. In addition, the spatial average is appended to the temporal cache. The second element x_2 has modality $m_2 = \text{text}$, which has no spatial dimension. We apply the text peripheral to x_2 and that output is sequentially appended to the temporal cache only. This goes on for all of the element in state s_t with only spatial modalities (images) being appended to the spatial cache. Spatially averaged peripheral outputs of all modalities are stored in the temporal cache. Algorithm 1 rigorously enumerates each step of this procedure.

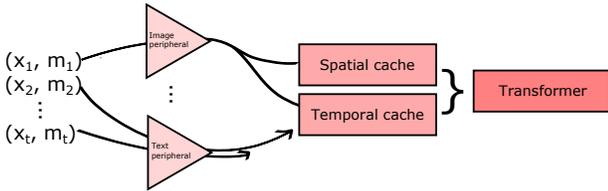


Figure 1: Diagram of our neural network architecture showing multimodal sequence elements passing through their respective peripherals, then being sequentially stored in the temporal and spatial caches of the transformer.

Algorithm 1: Storing peripheral outputs in spatial and temporal caches

Temporal cache = [], Spatial cache = []

$d_s = \text{spatial dimensions}$

for $t' = 1, 2, \dots, t$ **do**

$\tilde{x}_{t'} \leftarrow m_{t'} \text{ peripheral}(x_{t'}) \in \mathbb{R}^{d_s \times d_{\text{model}}}$

if $d_s > 1$ **do**

Spatial cache \leftarrow [Spatial cache, $\tilde{x}_{t'}$]

end if

Temporal cache \leftarrow [Temporal cache, $\frac{1}{d_s} \sum_{i=1}^{d_s} \tilde{x}_{t'}[i, :]$]

end for

3.2 Transformer

The temporal and spatial caches form the inputs into the transformer portion of the neural network. Figure 2 shows a schematic of OmniNet-based spatial-temporal transformer body with classifier and policy heads. First, the standard positional encoding [19] is added to the temporal cache before it enters the first multi-head attention block (with residual addition and layer normalization [2]). This first attention block's output form the 'queries', and along with the 'keys' and 'values' from the spatial cache, make up the inputs to the *gated* multi-head attention block [14]. In this way, temporal features can attend to spatial features and learn complementary cross-modality information. Furthermore, with gated multi-head attention, temporal attention scores respectively scale the corresponding attention scores of spatial cache elements. For

instance, if an image receives high attention in the first, temporal attention block, its corresponding subpixels will receive higher attention in this gated, spatial attention block. This is a mechanism to ensure high temporal attention translates to high spatial attention. We refer the reader to [14] for further details and discussion of gated multi-head attention blocks. Finally, the output of this gated, multi-head attention block (after another residual addition and layer normalization) form the inputs for two separate, feed forward heads: one for the policy and one for the classifier.

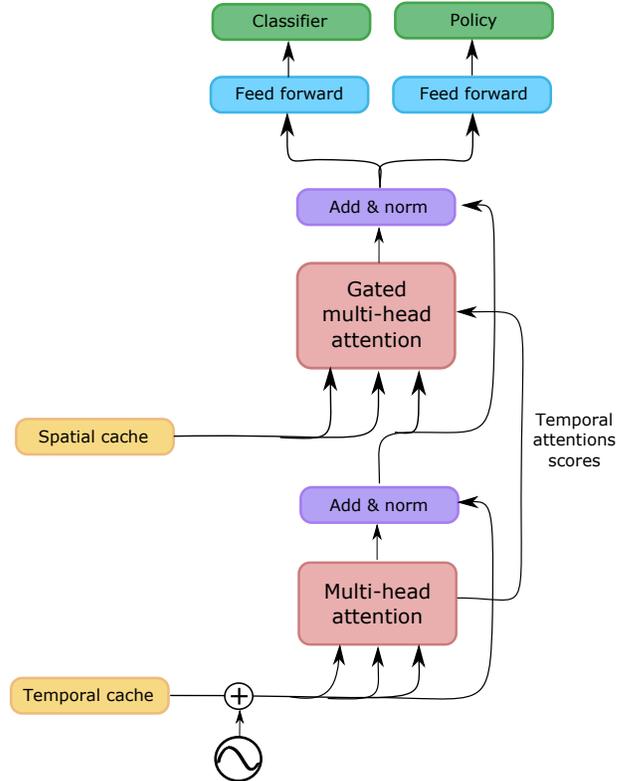


Figure 2: Diagram of spatial-temporal transformer body with classifier and policy heads.

4 CIS

Learning early classification is equivalent to maximizing the cumulative reward in (1). This cumulative reward can be reformulated as a function r depending on label y , classification prediction $\hat{y}_\alpha(\cdot|s_T)$, and classification time T given by $r(y, \hat{y}, T) = -\text{CE}(y, \hat{y}) - \mu T$. An important observation is that for a fixed \hat{y} and y this becomes a simple univariate function of time T . Utilizing this, CIS is able to learn (i) when to stop and classify and (ii) what classification to make in a more direct, supervised manner. First, CIS seeks to make the most accurate classification prediction at every time step. Second and concurrently, CIS learns the corresponding policy which yields the resulting optimized classification time. From this duality, CIS learns the ideal policy based off of its own classifications.

The loss function is given by $\mathcal{L}_{\text{CIS}} = \mathbb{E}_{\mathcal{X}} [\mathcal{L}_{\hat{y}} + \lambda \cdot \mathcal{L}_{\pi}]$ where

$$\begin{aligned} \mathcal{L}_{\hat{y}} &= \frac{1}{T_{\text{end}}} \sum_{t=1}^{T_{\text{end}}} \text{CE}(y, \hat{y}_{\alpha}(\cdot|s_t)) \\ \mathcal{L}_{\pi} &= \frac{1}{T_{\text{end}}} \sum_{t=1}^{T_{\text{end}}} \text{CE}(\tilde{\pi}_{\alpha}(\cdot|x, t), \pi_{\beta}(\cdot|s_t)) \\ \tilde{\pi}_{\alpha}(\cdot|x, t) &= \begin{cases} (1, 0) & \text{if } t < \tilde{T}_{\alpha}(x, y) \\ (0, 1) & \text{if } t \geq \tilde{T}_{\alpha}(x, y) \end{cases} \\ \tilde{T}_{\alpha}(x, y) &= \arg \max_t r(y, \hat{y}_{\alpha}(\cdot|s_t), t). \end{aligned}$$

Vector (1, 0) means ‘wait’ with probability 1 and (0, 1) is ‘stop and classify’ with probability 1. Scaling constant λ is a hyperparameter.

Unlike LARM, CIS does not rely on any help waiting for enough information; it is able to directly learn the optimal classification time in a supervised manner. During training $\tilde{\pi}_{\alpha}(\cdot|x, t)$ and \tilde{T}_{α} are calculated and treated as fixed labels per minibatch update. In inference, CIS simply takes the argmax action.

5 EXPERIMENTAL RESULTS

5.1 Datasets and Pareto Metric

Our first experiment is with the N24News Multimodal News Classification dataset [21]. It consists of New York Times news articles from different categories. Each article comprises five elements. In order of appearance, they are (i) headline, (ii) abstract, (iii) image, (iv) image caption, and (v) article body. We do not need to ingest all of the elements in an article to classify its category (economy, technology, etc.). Instead, we ingest element by element and classify the article after ingesting a minimal number of elements. We ingest the elements in the order they naturally appear in articles. To make the article body consistent in size with the other elements, however, we pad up or truncate down to 2,000 BERT [5] tokens and further divide it into 40 elements of 50 tokens. So each article’s sequence follows (headline, abstract, image, image caption, body 1, ..., body 40). The dataset contains 61,218 news articles in 24 well-balanced categories. We reserve a random 10% of articles to be the hold-out validation set, separate from the training set.

The second experiment is derived from the ESP Game dataset [20]. This dataset contains annotated, everyday images; each image is paired with a list of unique words describing that image. We can swap some pairings, so those images are no longer paired with their original list of words, and create the following task: Suppose you saw an image and then read the paired words one at a time. How quickly could you determine if the image and words were correctly or incorrectly paired (binary classification)? With many of the words being generic adjectives and nouns, like ‘blue’ or ‘person,’ the task is not trivial. We pad each word list up to 42 words, which is the largest such list, and order the words within each list by increasing uniqueness. This is done to match the design of the original ESP Game and trend of waiting longer for increasing information. The dataset contains 100,000 samples with correct and incorrect pairings evenly split. A random 10% of samples are reserved for the hold-out validation set. Here, cross-modality learning is explicit and necessary.

Our third and final experiment makes use of industry data and application. In this real use case, we have multimodal sequences composed of four elements: (i) structured categorical data, (ii) text, (iii) a bag of images, and (iv) another bag of images. For a given sample, each element arrives sequentially but in variable order. Associated with each sample is a binary label which we attempt to predict as accurately as possible, with as few elements received.

In reality, the features of the structured categorical data also arrive sequentially and with variable order. While we do not have access to these finer grained arrival time stamps, in consultation with subject matter experts, we mimic this process with the following procedure: We first train an XGBoost model [4] to classify samples’ structured data only. We can then identify no-importance features (feature importance scores of 0), low importance features (scores between 0 and 0.01), and high importance features (scores greater than 0.01). The structured data is artificially made to arrive three times. For the first arrival, we set the value of each feature to ‘missing’ with a probability according its importance. No-importance features are made ‘missing’ with 90% probability, low importance features with 95% probability, and high importance features at 99% probability. We encode ‘missing’ by adding a dimension to the one-hot encoding of categorical features. For the second arrival, we take the first arrival and replace ‘missing’ values with the true value with 20% probability. Similarly for the third arrival, we do the same on the second arrival. The first arrival replaces the original structured element’s sequence position. The second arrival is inserted two positions afterwards if possible, otherwise it immediately follows the first arrival. We do the same for inserting the third arrival after the second. For example, the most common sequence is (structured 1, text, structured 2, bag of images 1, structured 3, bag of images 2). In total, this dataset contains 63,030 samples with evenly split positive and negative labels. Again, we reserve a random 10% of samples to be the validation set.

To holistically compare LARM and CIS early classifiers, we construct their Pareto frontiers. In this way we can study the complete spectrum of each method’s accuracy-timeliness tradeoffs. For a specific μ , we evaluate the early classifier over the validation set and compute the mean classification time and accuracy after each training epoch. Sweeping over a range of μ values yields a set of accuracy-timeliness tradeoff points. Finally, the Pareto frontier emerges after removing all dominated points. Furthermore, we run three independent trials of this procedure to create three Pareto frontiers per method. The mean AUC of the Pareto frontiers is a holistic measure of the early classifier’s accuracy-timeliness trade-off capacity.

5.2 Implementation

For all three experiments, we sweep $\mu \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For CIS, we set the scaling constant $\lambda = 1$. The training set is optimized by using Adam with batch size 128 until validation accuracies and mean classification times plateau. All images are resized to 224×224 . If we pretrain a peripheral on the sequence labels, we write the accuracy in parentheses. For all peripherals, we explain the feature extractor and implement a single feed-forward layer of dimension d_{model} for the projector (explained in §3.1). For the spatial-temporal transformer, each multi-head attention block

has eight heads with queries, keys, and values of dimension 64. The classifier and policy heads’ feed-forward networks have one hidden layer of dimension 100 with ReLU activation.

For the N24News experiment, four separate BERT models are pretrained for each text source: Headline (73.6% acc.), abstract (78.8% acc.), caption (71.0% acc.), and body portions (69.1% acc.). We take the final hidden state of dimension 768 as the feature extraction. A ResNet-18 [8] model is pretrained on the images (48.6% acc.). The $7 \times 7 \times 512$ spatial layer before average spatial pooling is used as the feature extraction. Peripheral accuracies are in line with [21]. For the transformer network, $d_{\text{model}} = 500$ and the learning rate is 10^{-5} for CIS and 10^{-6} for LARM. Following [10], we keep LARM’s waiting parameter $\rho = 0.9$.

For the ESP Game image-words experiment, we cannot pretrain an image or words peripheral since both modalities are necessary to make an accurate classification. Accordingly, we simply utilize a ImageNet-pretrained ResNet-18 as the image peripheral where, again, the $7 \times 7 \times 512$ spatial layer before average spatial pooling is used as the feature extraction. For the words peripheral we use GloVe word embeddings [13] of dimension 300. For the transformer, $d_{\text{model}} = 300$ and the learning rate is 10^{-5} for CIS and 10^{-6} for LARM. LARM’s waiting parameter $\rho = 0.9$ again.

Finally, for our industry experiment, we do not apply a peripheral to the three structured data arrivals, just insert the projector introduced above. For reference though, a simple 500-dimensional single hidden-layer, feed-forward classifier yields a 64.8% accuracy for the first arrival, 78.1% accuracy for the second, and 80.1% for the third. We pretrain a BERT model on the last 512 tokens of the text data (67.1% acc.) and again take the final hidden state of dimension 768 as the feature extraction. For both bags of images, we pretrain separate ResNet-18 models where the average spatial pooling layer is also across images in a bag (53.6% and 53.0% accs.). Again, the $7 \times 7 \times 512$ spatial layers for each image before average spatial pooling is used as the feature extractions. For this transformer, $d_{\text{model}} = 500$ and the learning rate is 10^{-5} for both CIS and LARM. LARM waiting parameter $\rho = 0.9$ lead to poor results and lowering it to 0.5 yields the best performance.²

5.3 N24News Experiment

Figure 3 displays the Pareto frontiers for the N24News experiment. CIS’s mean AUC is 1.6% greater than LARM’s mean AUC. CIS slightly outperforms LARM, and we stress this is due to the supervised nature of the algorithm. We will see this performance gap grow as the data becomes more complex and the interplay between modalities more important in the following experiments.

5.4 ESP Game Image-Words Experiment

Figure 4 (top) displays the Pareto frontiers for the ESP Game image-words experiment. CIS holistically outperforms LARM. CIS’s mean AUC is 5.2% greater than LARM’s mean AUC. Reflected in this larger AUC margin, CIS is able to better capture the multimodal dependency.

To showcase CIS’s discerning patience, we investigate the distribution of stopping times compared to LARM. Figure 4 (bottom) shows just this using CIS and LARM with mean classification time

²We pledge to publish our code and add a link here upon acceptance of this paper.

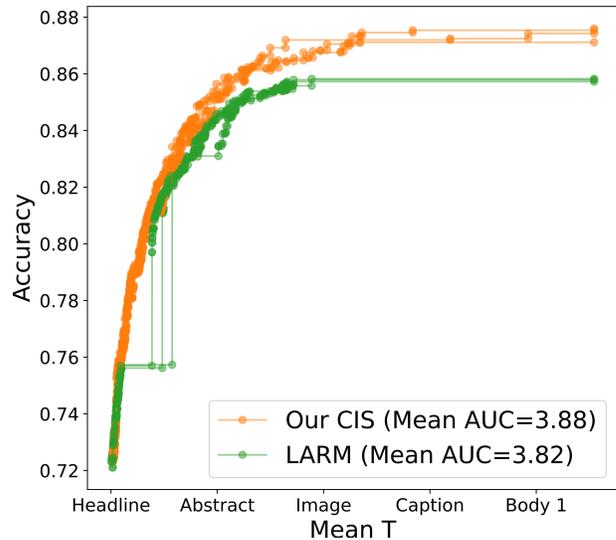


Figure 3: Pareto frontiers for the N24News experiment.

of 1.5 words (red circles in Figure 4 (top)). We can see that CIS (i) waits until at least the first word as that’s the minimum information needed to predict accurately and (ii) does not wait for as many words as LARM on the tail end. In other words, the distribution has lower spread.

5.5 Industry Experiment

Figure 5 (top) displays the Pareto frontiers for the industry experiment. Again, CIS holistically outperforms LARM. CIS’s mean AUC is 8.7% greater than LARM’s mean AUC.

We wish to study the stopping times for the variable modality arrivals. In Figure 5 (bottom), we show a Sankey plot of sequences and their respective stopping times for a specific CIS Pareto point (circled in red in Figure 5 (top)). The first observation is that CIS most often stops after receiving the second structured data (seen from ‘A’ markers). This of course makes sense since we are in the lower time penalty region and structured data is the most informative (highest accuracy peripheral). A second, more interesting observation is that CIS also frequently stops and classifies after receiving both the first structured data and text modality in that order (seen from ‘B’ markers). This suggests these two modalities have complementary information. As we can see, there is rich opportunity for studying early classification models.

6 CONCLUSION

Early classification has recently gained attention as an important adaptation of classification to dynamic environments. Methods like LARM and CIS represent the state-of-the-art. However, these methods have solely focused on unimodal sequences. To our knowledge, this paper is the first study of early classification of multimodal sequences. Not only do we stress the ubiquity of such problems in the real world but also demonstrate that an OmniNet-like spatial-temporal transformer combined with CIS is an effective approach.

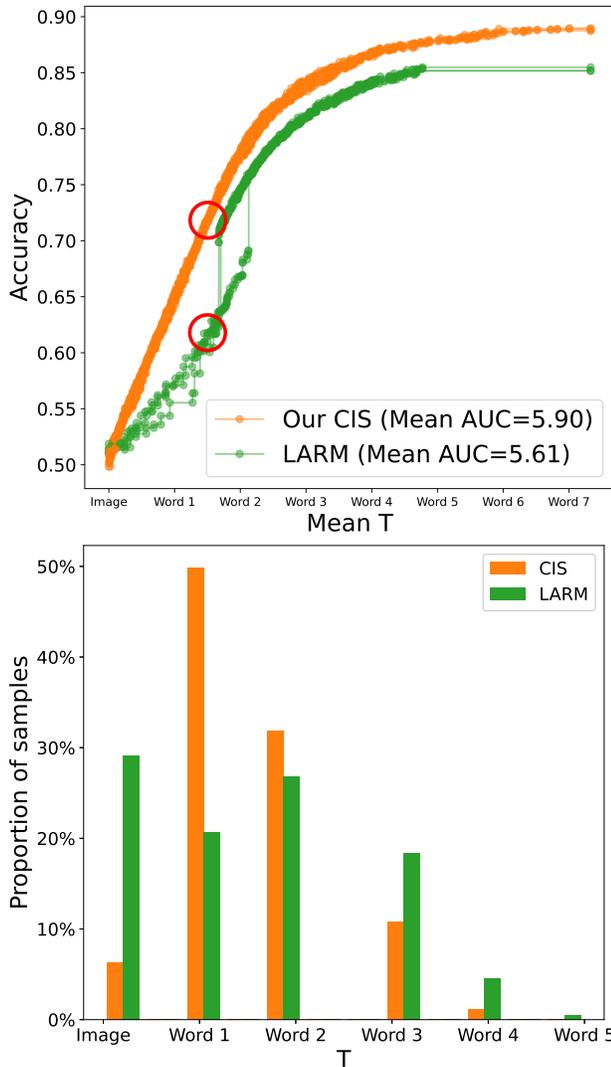


Figure 4: (Top) Pareto frontiers for the ESP Game image-words experiment. (Bottom) Histograms showing distribution of CIS and LARM classification times T .

For sure, multimodal sequences are an important extension of unimodal early classification.

REFERENCES

- [1] Ayush Agarwal, Ashima Yadav, and Dinesh Kumar Vishwakarma. 2019. Multimodal sentiment analysis via RNN variants. In *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)*. IEEE, 19–23.
- [2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
- [3] Alexander Cao, Jean Utke, and Diego Klabjan. 2023. A Policy for Early Sequence Classification. *arXiv preprint arXiv:2304.03463* (2023).
- [4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

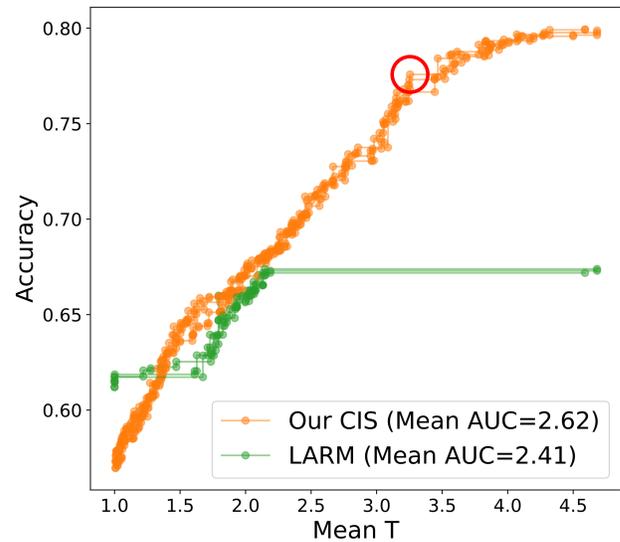


Figure 5: (Top) Pareto frontiers for the industry experiment. (Bottom) Sankey plot showing stopping times of samples with different modality arrivals.

- [6] Weijiang Feng, Naiyang Guan, Yuan Li, Xiang Zhang, and Zhigang Luo. 2017. Audio visual speech recognition with multimodal recurrent neural networks. In *2017 International Joint Conference on neural networks (IJCNN)*. IEEE, 681–688.
- [7] Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, and Elke Rundensteiner. 2019. Adaptive-halting policy network for early classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 101–110.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [10] Zhengjie Huang, Zi Ye, Shuangyin Li, and Rong Pan. 2017. Length adaptive recurrent model for text classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1019–1027.
- [11] Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2020. Finding decision jumps in text classification. *Neurocomputing* 371 (2020), 177–187.
- [12] Jiaxin Ma, Hao Tang, Wei-Long Zheng, and Bao-Liang Lu. 2019. Emotion recognition using multimodal residual LSTM network. In *Proceedings of the 27th ACM international conference on multimedia*. 176–183.
- [13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

- [14] Subhojeet Pramanik, Priyanka Agrawal, and Aman Hussain. 2019. Omninet: A unified architecture for multi-modal multi-task learning. *arXiv preprint arXiv:1907.07804* (2019).
- [15] Jimmy Ren, Yongtao Hu, Yu-Wing Tai, Chuan Wang, Li Xu, Wenxiu Sun, and Qiong Yan. 2016. Look, listen and learn—A multimodal LSTM for speaker identification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.
- [16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- [17] Hao Tang, Wei Liu, Wei-Long Zheng, and Bao-Liang Lu. 2017. Multimodal emotion recognition using deep neural networks. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part IV 24*. Springer, 811–819.
- [18] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, Vol. 2019. NIH Public Access, 6558.
- [19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [20] Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 319–326.
- [21] Zhen Wang, X Shan, Xiangxie Zhang, and J Yang. 2022. N24News: A New Dataset for Multimodal News Classification. In *2022 Language Resources and Evaluation Conference, LREC 2022*. European Language Resources Association (ELRA).
- [22] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning* (1992), 5–32.
- [23] Peng Xu, Xiatian Zhu, and David A Clifton. 2022. Multimodal learning with transformers: A survey. *arXiv preprint arXiv:2206.06488* (2022).
- [24] Jun Yu, Jing Li, Zhou Yu, and Qingming Huang. 2019. Multimodal transformer with multi-view visual representation for image captioning. *IEEE transactions on circuits and systems for video technology* 30, 12 (2019), 4467–4480.