NORA.ai Summer Research School Multi-modal learning course

Anonymous Home Exam
Candidate Number 8

1 Problem 1

To solve Problem 1 of the assignment attached code was used. It consists of four files, `image_model.py`, `audio_model.py`, `multimodal_model.py` each contain the data processing and model definitions for the corresponding parts of the assignment and `main.ipynb` is a jupyter notebook that imports from the `.py` files and performs training and subsequent visualisation of results. Neural networks were implemented and trained using pytorch lightning for easier readability, training was monitored using Weights&Biases. Even though the data was initially only split into training and testing datasets, we decided to go with the train/validation/test splitting approach in order to avoid having to train for a fixed number of epochs or using the test dataset when selecting the optimal epoch. Therefore 20% of the original training dataset was designated as validation set and early stopping callback was used to select model weights from the epoch that minimized validation loss. This means that in figures 1, 5, 9, 11 the plotted values of validation accuracy correspond to this validation set and the value of accuracy on the test dataset is only reported for the final model.

AI coding assistant tools (Copilot) were used to help with code implementation and documentation.

1.1 1a

For this task we designed and implemented a simple unimodal classification convolutional neural network to only work with the image data. It is implemented in `image_model.py` as Conv2Net pytorch module. The network consists of two 2D convolutional layers, each of which is followed by a ReLU activation and a max-pooling layer. Both convolutional layers have a kernel size of 3, the first one has 32 output channels (filters), the second one has 64 output channels. The kernel size for both of the max-pooling layers is 2 with a stride of 2. After the images pass through this "featurizing" part of the network, they are processed by two linear fully connected layers with output sizes 128 and 10, with another ReLU activation after the first one. No training data augmentation methods were used, instead dropout was used to prevent overfitting.

On figure 1, selected metrics from the training process are displayed. As mentioned before, Early stopping callback was used to control the number of epochs with patience parameter set to 5. In the end epoch 26 was selected as the best one and weights from that epoch were used further. Validation loss and accuracy are relatively stable throughout the training, the fluctuations in training loss are caused by a high rate of dropout, which succeeds in preventing early overfitting.

Model's evaluation on test dataset leads to Test loss of 0.437 and Test accuracy of 88.2%. Classification results are displayed as a confusion matrix on figure 2. The most significant non-diagonal entries suggest that the model most often mistakens number pairs such as 3-5, 4-9 or 3-8, which is in accordance with naive expectations.

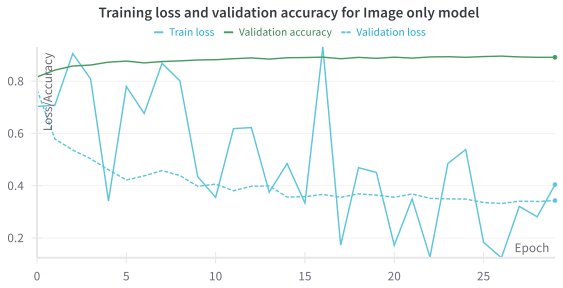

Figure 1. Training loss and Validation accuracy throughout the training of Image only model.

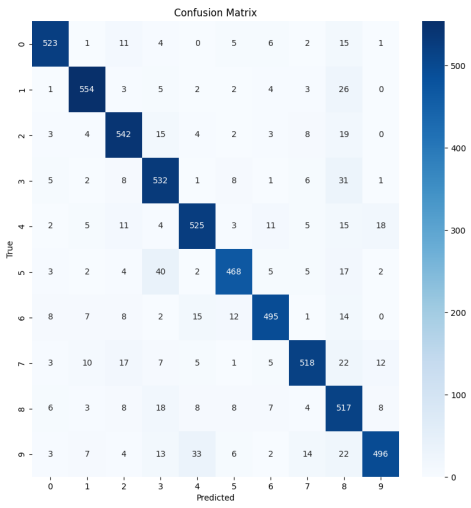

Figure 2. Confusion matrix on Test dataset for Image only model

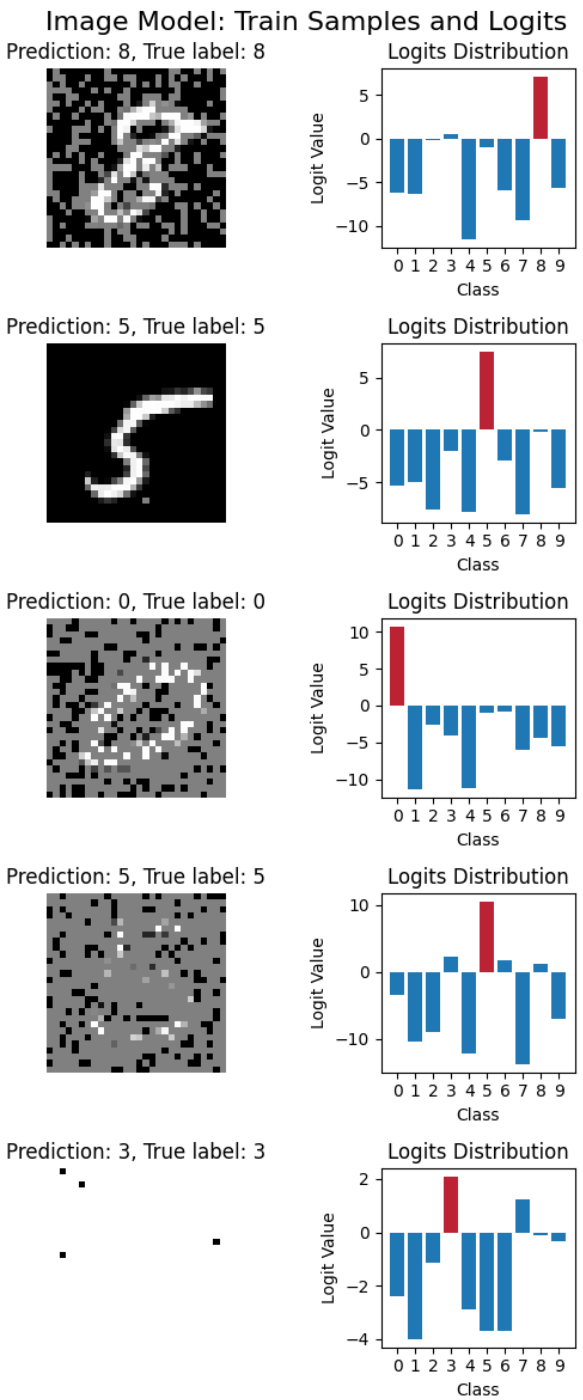

Figure 3. Five randomly sampled images from the training dataset with logits from the Image only model.

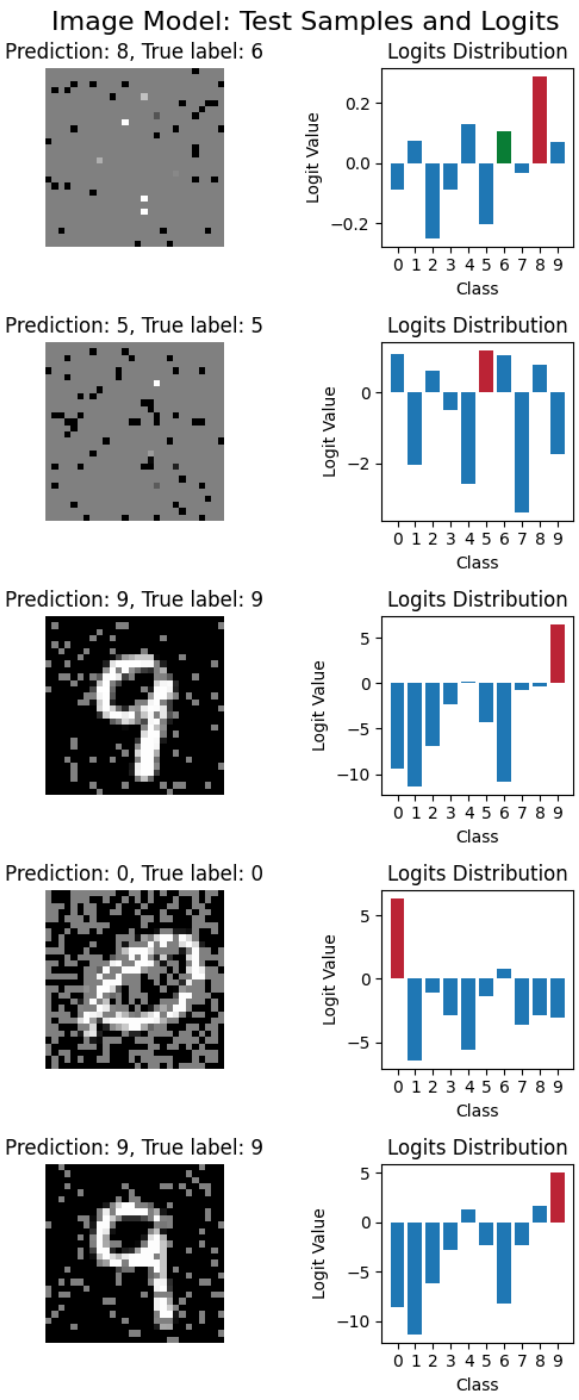

Figure 4. Five randomly sampled images from the test dataset with logits from the Image only model.

On figures 3 and 4, there are 5 randomly selected images from training and testing datasets along with their logits from the trained network. We can see that the amount of noise in images is sometimes incredibly high, rendering the classification task basically impossible, such as for the last image from training dataset - it is likely that the network is overfitted and "remembers" this specific image, especially because the logits show relatively high confidence in the correctly predicted class. On the other hand, in case of the second image from the testing dataset, we can see quite high uncertainty of the model with its correct prediction. Considering this sample would probably not be correctly classified by a person, there seems to be a decent degree of generalization in the model.

1.2 1b

For the second task we generally followed the same approach as in the first task. The main difference was obviously the shape of the data to work with, which dictated the use of a different neural network architecture. Initially, we experimented with InceptionTime neural network [1], which showed great results on unrelated time-series processing, however due to the different properties of our audio samples it turned out to be unnecessarily complex and computationally inefficient.

Rather, we ended up using a simple convolutional network, implemented as Conv1Net in `audio_model.py`. The featurizer consists of two 1D convolutional layers followed by ReLU activations. The two convolutional layers are both "same" padded to prevent the shortening of a sequence due to filter overlap, but they both use stride of four, effectively quartering the temporal size of the data (therefore no max-pooling is necessary). Kernel length is 100 and 25, while number of output channels is 32 and 64 for the two layers respectively. The second part of the network has a similar setup as in Conv2Net - two fully connected layers of output sizes 128 and 10 with a ReLU activation in between. Again, dropout was used as a regularization method with no data augmentations.

In figure 5, the metrics collected during training are displayed. The setup of Early stopping remained the same with patience parameter of 5. The optimal epoch in this case turned out to be already epoch 6, after which we observe noticeable overfitting as Validation loss does not follow the decreasing trend of Training loss. Training loss seems to fluctuate a bit less than in figure 1 for an image only model, even though the same dropout rate of 0.5 was used. Potentially, stronger regularization could be beneficial. Nevertheless, the Validation accuracy remains stable, hovering over 10 percentage points lower than in case of image only model.

The evaluation on test dataset yielded Test loss of 0.923 and Test accuracy of 71.5%. Classification results are displayed as a confusion matrix in figure 6. It is difficult to interpret or directly compare this result to the results of image only model, as we cannot asses the levels of noise present in either modality. Based on the confusion matrix the most difficult to distinguish digit pairs are 3-4 and 1-9. In this case, naive intuition falls short of explaining this behavior, as these digits do not seem to sounds alike.

Similarly as for previous task, in images 7 and 8, there are 5 randomly chosen samples from training and testing datasets along with their logits from the trained audio only network. Unfortunately, the plot of a raw audio waveform is much less informative for a human eye, than an image was in previous part. Nevertheless, we can see one extremely distorted sample (second one from training dataset) that is classified with high certainty, which suggests overfitting on this difficult sample.

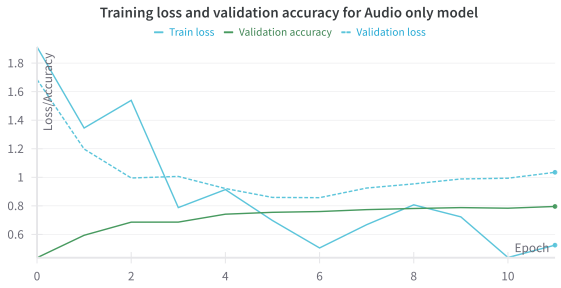

Figure 5. Training loss and Validation accuracy throughout the training of Audio only model.

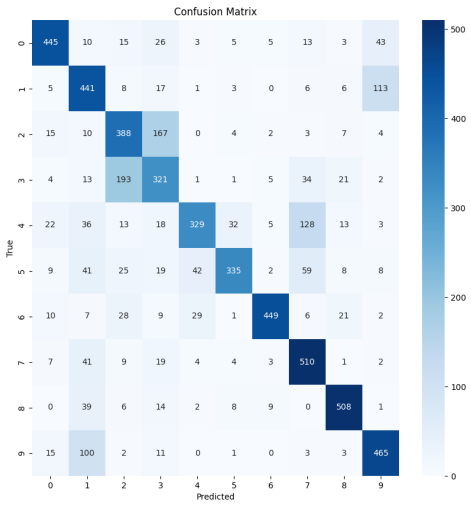

Figure 6. Confusion matrix on Test dataset for Audio only model

1.3 1c

When comparing the individual modality approaches, it is clear that the image modality has more discriminative power as the image only model reached Test accuracy of 88.2% compared to 71.5% for the audio only model. The classification network for audio modality could likely be improved to achieve better performance by leveraging some pretrained audio encoders, but we decided to keep the models small, local and easy to read, because, as was mentioned before, it is not clear if the two modalities have been corrupted by noise to a similar degree.

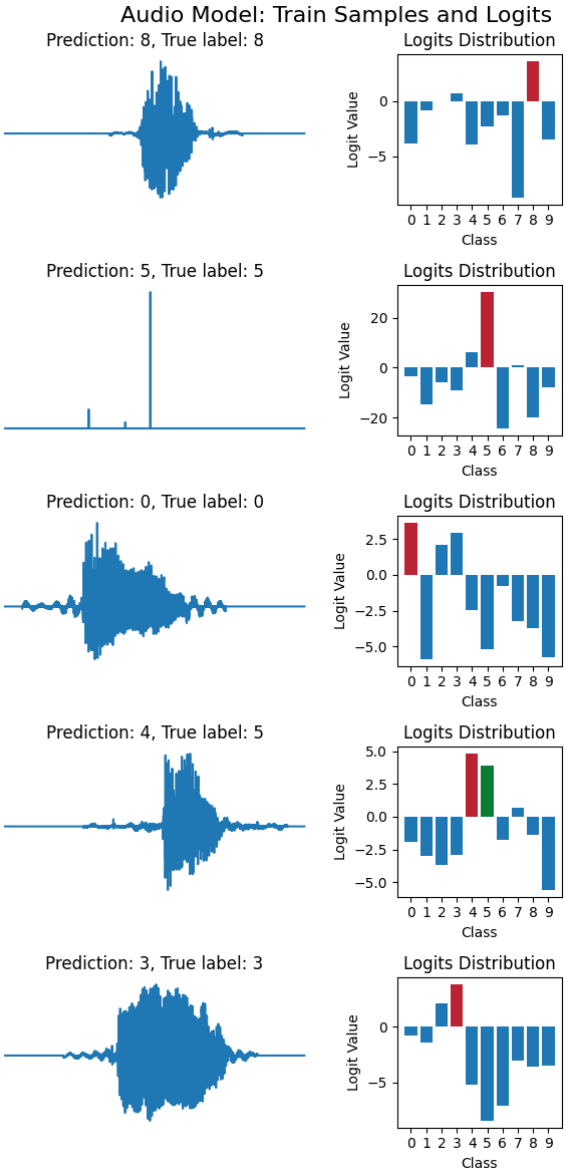

Figure 7. Five randomly sampled audio waves from the training dataset with logits from the Audio only model.

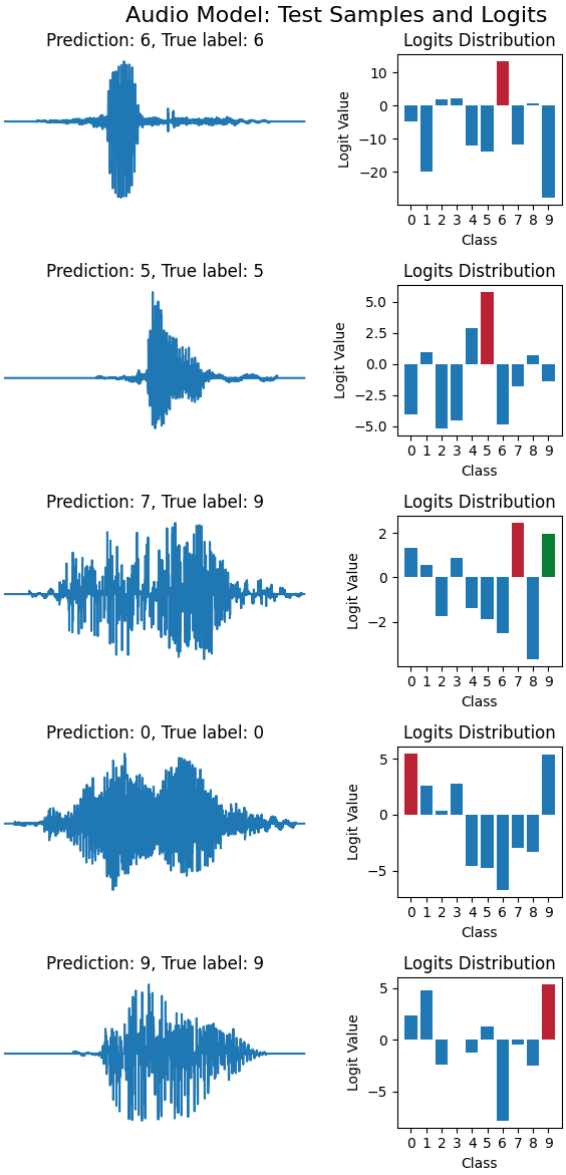

Figure 8. Five randomly sampled audio waves from the test dataset with logits from the Audio only model.

1.4 1d

Assuming, that the image and audio captioning errors are independent, we can expect that a large portion of samples with corrupted image will have a healthy audio recording and vice versa. In that case, it will likely be beneficial to use both of the modalities, rather than relying on a single one, as the other can act as backup in case of captioning error of the first one.

Multimodal learning enables us to do just this - use both of the modalities of a sample as separate inputs into two modality specific featurizers that process them and embed them in modality specific latent spaces. It can be beneficial for the two latent spaces to have the same dimensions, as the next step in multimodal pipeline - the fusion of the two modalities, might require it, depending on the specific fusion approach that is used. Using the nomenclature given in [2], this methodology can be specified as early fusion with abstract modalities. In a fully supervised multi-modal classification setup, which is our case, a standard classification head can be used on the fused representation of the sample, with a categorical cross entropy as loss function.

The fusion itself can be performed in a number of ways. Assuming we have the image inner features $f_{img} \in \mathbb{R}^{n_i}$ and the audio inner features $f_{aud} \in \mathbb{R}^{n_a}$ of a given sample, the simplest and perhaps the most common one is simple concatenation of features from individual modalities. One of the benefits is that it does not require the different modality embeddings to have identical dimensions. Further, we describe other ways to fuse f_{img}, f_{aud}.

Concatenation fusion

$$f_{fused} = [f_{img}, f_{aud}]$$

Element-wise operations
For all of the element-wise fusion methods the dimensions must match, i.e. $n_i = n_a$. Unlike for the other fusion methods, these ones have no learnable parts.

- **Addition fusion**

$$f_{fused} = f_{img} + f_{aud}$$

- **Multiplication fusion**

$$f_{fused} = f_{img} \odot f_{aud}$$

- **Maximum fusion**

$$f_{fused} = \max\left(f_{img}, f_{aud}\right)$$

Attention-based fusion
Uses bidirectional cross-modal multihead attention mechanism. Image features attend to audio features and vice versa using the same MultiheadAttention module, then the attended features are element-wise added and layer normalized. Again, $n_i = n_a$.

$$f_{fused} = \text{LayerNorm}\Big(\text{MhA}\left(f_{img}, f_{aud}, f_{aud}\right) + \text{MhA}\left(f_{aud}, f_{img}, f_{img}\right)\Big)$$

Gated fusion
Multiplicative sigmoidal gates are learned for each modality to control contribution of each feature into the final representation. Gated features are concatenated and then gated again, using another learned gate. Again, $n_i = n_a$. Used in [3].

$$f_0 = \left[\sigma\left(f_{img} * W_{img}\right) \odot f_{img}, \sigma\left(f_{aud} * W_{aud}\right) \odot f_{aud}\right]$$

$$f_{fused} = \sigma\left(f_0 * W_0\right) \odot f_0$$

Bilinear fusion
This fusion method computes an outer product between feature vectors, thus capturing all pairwise interactions. The dimensions of modality embedding do not have to match and we can specify the desired fused dimension. Let $W \in \mathbb{R}^{n_f \times n_i \times n_a}$.

$$f_{fused_i} = f_{img}^T * W_i * f_{aud}$$

Factorized Bilinear fusion
A more efficient version of bilinear fusion, that factorizes the bilinear interaction matrix [4]. Projects features to common hidden space before computing interactions. Does not require $n_i = n_a$ and allows the choice of output dimension n_f. $W_{img} \in \mathbb{R}^{n_i \times h}$, $W_{aud} \in \mathbb{R}^{n_a \times h}$, $W_f \in \mathbb{R}^{h \times n_f}$.

$$f_{fused} = \Big(\left(f_{img} * W_{img}\right) \odot \left(f_{aud} * W_{aud}\right)\Big) * W_f$$

1.5 1e

In this section we reused the components that we created for unimodal tasks. We used the Conv2Net and Conv1Net to process the image and audio modalities of a given sample. Rather than obtaining predictions from the two models independently and performing late fusion, we took the inner representations from both networks after the second-to-last layer (128 features) and performed early fusion and subsequent classification with fused classification head. The classification head consists of two fully connected layers with output sizes of 64 and 10 with a ReLU activation in between. In file `multimodal_model.py`, this model is implemented along with all the multimodal fusion methods described in previous section. Here we are using the simplest Concatenation Fusion method. All of the weights in Conv2Net and Conv1Net were initialized randomly and trained from scratch for the multimodal model.

In figure 9, the evolution of losses and validation accuracy are displayed. Just as before, Early stopping with patience of 5 epoch was used to identify the optimal epoch, in this case epoch 16. We do not observe any significant overfitting, the fluctuations in Training loss can be attributed to dropout with rate 0.5 used in both image and audio branches of the network. The Validation accuracy is stable and hovers around 95%.

When evaluated on test dataset we received final metrics of 0.278 for Test loss and 94.5% for Test accuracy. In figure 10 the confusion matrix of predictions for the test dataset is displayed. Apparently the most confusing digit pair for this setup is 2-3. When compared with the confusion matrix for image only model in figure 2 we can see worse performance for this specific digit pair, however compared to the confusion matrix for audio only model in figure 6, where 2-3 is also the most confusing digit pair, we observe a significant improvement with multimodal model. Overall, the performance of the multimodal model is much better than both of the single modality models.

1.6 1f

In this task we trained the multimodal model with all the other fusion methods described in previous

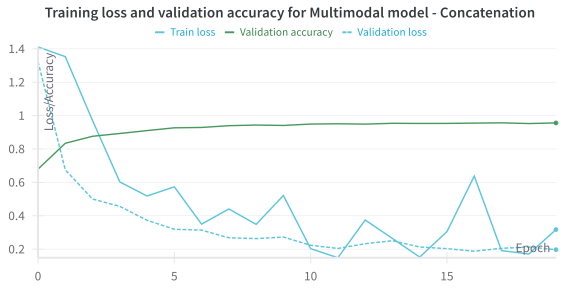

Figure 9. Training loss and Validation accuracy throughout the training of Multimodal model with concatenation fusion.

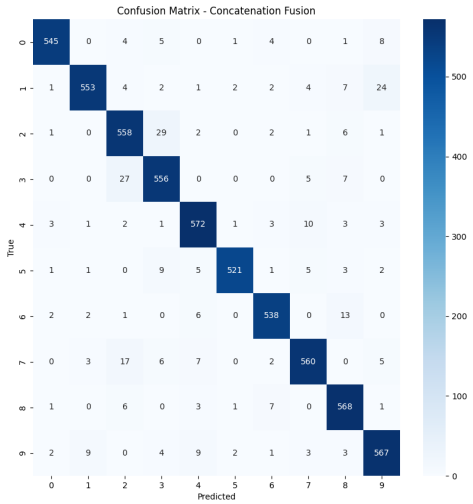

Figure 10. Confusion matrix on Test dataset for Multimodal model with concatenation fusion

sections. For simplicity and brevity, we only provide the detailed description of training process for the Attention-based fusion, for other methods only the final metrics on test dataset are reported.

In figure 11, the losses and Validation accuracy during training of multimodal model with attention-based fusion are displayed. We do not observe any meaningful difference compared to Concatenation fusion model, except for lower fluctuations of Training loss. This might be caused by the fact that the model has additional parameters in the attention fusion module which help combat the instabilities caused by dropout in other parts of the network.

The confusion matrix of predictions on the test dataset for attention-based fusion model, displayed in figure 12, does not reveal any significant changes compared to concatenation fusion model. The performance on the digit pair 2-3 has slightly improved.

Final metrics on test dataset for the attention-based fusion model, as well as for all the other fusion strategies, can be found in table 1. Multiplication, Maximum and Bilinear strategies fall short of the others and should not be used in this setting. It is worth mentioning that the attention-based fusion model reached the lowest Test loss, however the performance of Concatenation, Addition, Gated and Factorized Bilinear strategies is very similar and there is no definitive best option for this problem setup.

Fusion method	Test loss	Test accuracy
Concatenation	0.278	94.5%
Addition	0.276	**94.9%**
Multiplication	0.464	88.9%
Maximum	0.312	92.7%
Attention	**0.210**	94.0%
Gated	0.237	94.0%
Bilinear	0.449	86.0%
Factorized Bilinear	0.213	94.0%

Table 1. Test dataset metrics for multimodal model with different fusion strategies.

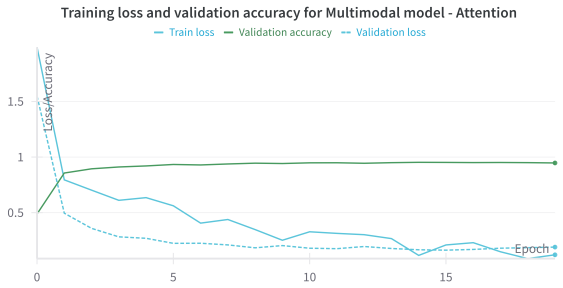

Figure 11. Training loss and Validation accuracy throughout the training of Multimodal model with attention fusion.

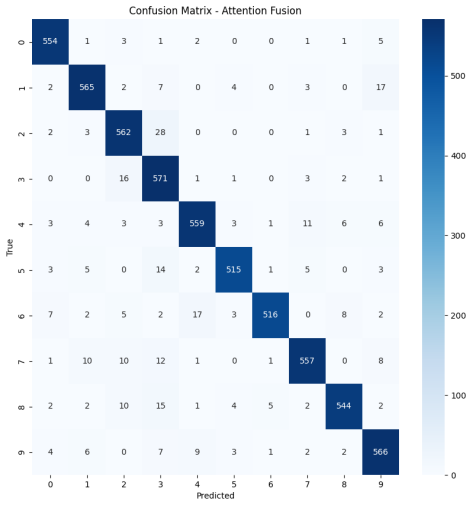

Figure 12. Confusion matrix on Test dataset for Multimodal model with attention fusion

2 Problem 2

2.1 2a

Assuming a negative loss function, the formula for NT-Xent for our case, where we only have one positive and one negative sample with respect to a single anchor sample $\mathbf{u}$, can be rewritten as

$$\text{NT-Xent} = \log\left(\frac{\exp\left(\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)/\tau\right)}{\sum_{\boldsymbol{v}\in\{\boldsymbol{v}^+,\boldsymbol{v}^-\}}\exp\left(\text{sim}(\boldsymbol{u},\boldsymbol{v})/\tau\right)}\right).$$

Then, applying the logarithm onto the factorized fraction, we get the desired output

$$\text{NT-Xent} = \frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau} - \log\left(\sum_{\boldsymbol{v}\in\{\boldsymbol{v}^+,\boldsymbol{v}^-\}}\exp\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v})}{\tau}\right)\right). \quad (1)$$

2.2 2b

Substituting $Z(\boldsymbol{u})$ into equation (1) we get a shortened expression for the loss

$$\text{NT-Xent} = \frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau} - \log\left(Z(\boldsymbol{u})\right).$$

Then, the gradient of the loss with respect to $\boldsymbol{u}$ can be written as

$$\frac{\partial\text{NT-Xent}}{\partial\boldsymbol{u}} = \frac{\boldsymbol{v}^+}{\tau} - \frac{Z'(\boldsymbol{u})}{Z(\boldsymbol{u})},$$

where

$$Z'(\boldsymbol{u}) = \frac{\boldsymbol{v}^+}{\tau}\exp\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau}\right) + \frac{\boldsymbol{v}^-}{\tau}\exp\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}{\tau}\right).$$

Collecting the common factors we get the final expression

$$\frac{\partial\text{NT-Xent}}{\partial\boldsymbol{u}} = \frac{\boldsymbol{v}^+}{\tau}\left(1 - \frac{\exp\left(\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)/\tau\right)}{Z(\boldsymbol{u})}\right) - \frac{\boldsymbol{v}^-}{\tau}\left(\frac{\exp\left(\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)/\tau\right)}{Z(\boldsymbol{u})}\right).$$

We disregard the sum over $\boldsymbol{v}^-$ in the final expression, since there is only one negative sample in our setting.

2.3 2c

To process the gradient of the logistic loss function it is useful to generally express the derivative of a sigmoid function $\sigma(x) = 1/(1+\exp(-x))$

$$\sigma'(x) = \frac{\exp(-x)}{\left(1+\exp(-x)\right)^2} = \frac{1}{1+\exp(-x)}\frac{1}{\frac{1}{\exp(-x)}+1} = \sigma(x)\sigma(-x). \quad (2)$$

Then, when deriving the logistic loss function we get

$$\frac{\partial\text{NT-Log}}{\partial\boldsymbol{u}} = \frac{\sigma'\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau}\right)\frac{\boldsymbol{v}^+}{\tau}}{\sigma\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau}\right)} + \frac{\sigma'\left(-\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}{\tau}\right)\left(-\frac{\boldsymbol{v}^-}{\tau}\right)}{\sigma\left(-\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}{\tau}\right)},$$

and using the equation (2) we can simplify it to the final form

$$\frac{\partial\text{NT-Log}}{\partial\boldsymbol{u}} = \sigma\left(-\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}{\tau}\right)\frac{\boldsymbol{v}^+}{\tau} - \sigma\left(\frac{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}{\tau}\right)\frac{\boldsymbol{v}^-}{\tau}.$$

When focusing on use in our scenario - only having one positive and one negative samples, both loss functions are set up in a way that they promote high similarity between $\boldsymbol{u}$ and $\boldsymbol{v}^+$ while penalizing similarity between $\boldsymbol{u}$ and $\boldsymbol{v}^-$. The main difference between them is that in NT-Xent the factor weighing the contribution to the gradient coming from $\boldsymbol{v}^-$ is also dependent on the positive sample (and on other negative samples when working with more than one - effectively being a softmax), while in NT-Log the factor is calculated separately and is only dependent on the similarity between $\boldsymbol{u}$ and $\boldsymbol{v}^-$. If we consider $\tau = 1$ for simplicity, these factors can be expressed as

$$f_{Log} = -\sigma\left(\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)\right),$$

$$f_{Xent} = -\frac{e^{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}}{e^{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)}+e^{\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}} = -\frac{1}{1+e^{\text{sim}(\boldsymbol{u},\boldsymbol{v}^+)-\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)}} = -\sigma\left(\text{sim}(\boldsymbol{u},\boldsymbol{v}^-) - \text{sim}(\boldsymbol{u},\boldsymbol{v}^+)\right).$$

In table 2 the values of f_{Xent} and f_{Log} are calculated for selected combinations of values of similarities between $\boldsymbol{u}$ and $\boldsymbol{v}^+$, $\boldsymbol{v}^-$. As we can see, the value of f_{Xent} converges to the values of f_{Log} for $\text{sim}(\boldsymbol{u},\boldsymbol{v}^+) \to 0$. Also $f_{Xent} > f_{Log}$ for $\text{sim}(\boldsymbol{u},\boldsymbol{v}^+) > 0$ and vice versa.

For both losses it is true that the "harder" the negative sample (the higher $\text{sim}(\boldsymbol{u},\boldsymbol{v}^-)$ is), the lower the coefficient weighing the negative samples direction in the gradient is. However NT-Xent loss also

$\text{sim}(u, v^+)$	$\text{sim}(u, v^-)$	f_{Xent}	f_{Log}
0.99	0.5	−0.380	−0.622
0.5	0.5	−0.5	−0.622
0.01	0.5	−0.620	−0.622
-0.5	0.5	−0.731	−0.622
0.99	-0.5	−0.184	−0.378
0.5	-0.5	−0.269	−0.378
0.01	-0.5	−0.375	−0.378
-0.5	-0.5	−0.5	−0.378

Table 2. Values of f_{Xent} and f_{Log} for various similarity scores between $\boldsymbol{u}$, $\boldsymbol{v}^+$ and $\boldsymbol{v}^-$.

takes into account how well the anchor sample is aligned with the positive sample and for a fixed value of $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^-)$, the better aligned they are (the higher $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^+)$ is) the higher the values of f_{Xent} is. In a setting with multiple negative samples, this translates to the fact that the gradient of NT-Xent is repulsed more from the negative samples that have higher similarity with $\boldsymbol{u}$ relative to other ones.

2.4 2d

The temperature parameter τ scales the similarity in both loss functions - this affects its sharpness.

τ very small ($\tau \to 0$) When the similarities are divided by a very small number, they become very large in absolute value before going into either loss function.

For NT-Xent this results in the loss being either almost zero for $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^+) > \text{sim}(\boldsymbol{u}, \boldsymbol{v}^-)$ or very negative for $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^+) < \text{sim}(\boldsymbol{u}, \boldsymbol{v}^-)$. This means that there would only be a reasonable gradient and therefore learning in case that the negative sample is more similar to the anchor than the positive sample. In a setting with multiple negative samples the softmax in the loss would become very sharp and behave like a hard classification, only favoring the most similar sample - gradient can become sparse or unstable.

For NT-Log the otputs of sigmoids become near 0 or 1 depending on the sign of the similarity. This means that the only meaningful contributions to a nonzero loss happen when $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^+) < 0$ for the positive sample or $\text{sim}(\boldsymbol{u}, \boldsymbol{v}^-) > 0$ for the negative sample. This might encourage a strong separation with the "hard" negative samples at the beginning of training, but will likely lead to gradient vanishing and training stagnation.

τ very large ($\tau \to \infty$) When the similarities are divided by a very large number, they become very small in absolute value before going into either loss function.

In NT-Xent, this leads to the first term in equation (1) to approach zero, while the argument of logarithm in the second term converges to two (N in case of more negative samples), so $\lim_{\tau \to \infty} \text{NT-Xent} = -\log 2$. Also, the gradient is minimal as its magnitude follows τ^{-1}, which leads to stagnant, unfocused training.

In NT-Log, as both arguments of sigmoids converge to zero with increasing τ, we get $\lim_{\tau \to \infty} \text{NT-Log} = 2\log(0.5)$. As with the previous loss function, both the factors weighing the contribution from $\boldsymbol{v}^+, \boldsymbol{v}^-$ in the gradient go to zero, which prevents the model from learning useful representations.

References

[1] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. "InceptionTime: Finding AlexNet for time series classification". In: *Data Mining and Knowledge Discovery* 34.6 (Sept. 2020), pp. 1936–1962. ISSN: 1573-756X. DOI: 10.1007/s10618-020-00710-y. URL: http://dx.doi.org/10.1007/s10618-020-00710-y.

[2] P. P. Liang, A. Zadeh, and L.-P. Morency. *Foundations and Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions*. 2023. arXiv: 2209.03430 [cs.LG]. URL: https://arxiv.org/abs/2209.03430.

[3] J. Arevalo, T. Solorio, M. Montes-y-Gómez, and F. A. González. *Gated Multimodal Units for Information Fusion*. 2017. arXiv: 1702.01992 [stat.ML]. URL: https://arxiv.org/abs/1702.01992.

[4] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. *Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding*. 2016. arXiv: 1606.01847 [cs.CV]. URL: https://arxiv.org/abs/1606.01847.