This is the supplementary material for **Heterogeneous Model Reuse via Optimizing Multiparty Multiclass Margin**.

Appendix A: Proof

Here we provide the proof of Theorem 1 presented in main paper. Let us restate the theorem first.

Suppose there are two parties A and B equipped with linear predictors defined by w_A and w_B. Assume $\mathcal{Y}_A = \mathcal{Y}_B = \mathcal{Y}$, the calibration operation on received example (x, y, y^-) is:

$$\begin{aligned} w_{i^+}^{(t+1)} &= w_{i^+}^{(t)} + \eta\Psi(x, y), \\ w_{i^-}^{(t+1)} &= w_{i^-}^{(t)} - \eta\Psi(x, y^-), \end{aligned} \tag{1}$$

where $\eta > 0$ controls the step size.

Theorem 1. *Assume $\|x\| = 1$, then the calibration operation described in* (1) *on linear predictors $\{h_A, h_B\}$ defined by $\{w_A, w_B\}$ will increase the MPMC-margin on sent example (x, y) by at least η.*

Proof. Our goal is to show that after one calibration operation at iteration t,

$$\rho_H^{(t+1)}(x, y) - \rho_H^{(t)}(x, y) \geq \eta. \tag{2}$$

We prove this theorem by enumerating all possible cases. For a selected non-positive margin example (x, y) at iteration t, four cases may happen:

$$\begin{aligned} \text{I: } & i^+ = A, i^- = A; \\ \text{II: } & i^+ = A, i^- = B; \\ \text{III: } & i^+ = B, i^- = A; \\ \text{IV: } & i^+ = B, i^- = B. \end{aligned}$$

Without loss of generality, it suffices to prove that (2) holds in case I and II. The last two cases can be proved similarly due to the symmetry between two parties.

If the algorithm enters case I at iteration t, according to the definition of MPMC-margin, we have

$$\rho_H^{(t)}(x, y) = \langle w_A^{(t)}, \Psi(x, y)\rangle - \langle w_A^{(t)}, \Psi(x, y^-)\rangle \leq 0. \tag{3}$$

Besides, case I requires that h_A makes both maximum predictions on the correct class y and the incorrect class y^- between h_A and h_B, so we can get the following inequalities:

$$\langle w_A^{(t)}, \Psi(x, y)\rangle \geq \langle w_B^{(t)}, \Psi(x, y)\rangle, \tag{4}$$

$$\langle w_A^{(t)}, \Psi(x, y^-)\rangle \geq \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_A^{(t)}, \Psi(x, y')\rangle, \tag{5}$$

$$\langle w_A^{(t)}, \Psi(x, y^-)\rangle \geq \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_B^{(t)}, \Psi(x, y')\rangle. \tag{6}$$

The calibration operation as described in (1) now sequentially updates w_A by:

$$w_A^{(t+1)} = w_A^{(t)} + \eta\big(\Psi(x, y) - \Psi(x, y^-)\big). \tag{7}$$

Now we check how the MPMC-margin changes at $t + 1$. Notice that $\|\Psi(x, y)\| = \|x\| = 1$ and $\langle\Psi(x, y), \Psi(x, y^-)\rangle = 0$, the prediction value on the correct class y is increased because

$$\begin{aligned} &\langle w_A^{(t+1)}, \Psi(x, y)\rangle - \langle w_A^{(t)}, \Psi(x, y)\rangle \\ =&\langle w_A^{(t+1)} - w_A^{(t)}, \Psi(x, y)\rangle \\ =&\eta\langle\Psi(x, y) - \Psi(x, y^-), \Psi(x, y)\rangle \\ =&\eta\big(\|\Psi(x, y)\|^2 - \langle\Psi(x, y), \Psi(x, y^-)\rangle\big) \\ =&\eta(1 - 0) = \eta. \end{aligned} \tag{8}$$

Combine (4) and (8), and recall that w_B stays the same,

$$\langle w_A^{(t+1)}, \Psi(x,y)\rangle > \langle w_A^{(t)}, \Psi(x,y)\rangle \geq \langle w_B^{(t)}, \Psi(x,y)\rangle = \langle w_B^{(t+1)}, \Psi(x,y)\rangle. \tag{9}$$

Therefore $\langle w_A^{(t+1)}, \Psi(x,y)\rangle$ remains the maximum prediction on y at $t+1$, and will be used in computing $\rho_H^{(t+1)}(x,y)$. Then for either $i^- \in \{A, B\}$,

$$\begin{aligned}
&\rho_H^{(t+1)}(x,y) - \rho_H^{(t)}(x,y) \\
=&\big(\langle w_A^{(t+1)}, \Psi(x,y)\rangle - \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_{i^-}^{(t+1)}, \Psi(x,y')\rangle\big) - \big(\langle w_A^{(t)}, \Psi(x,y)\rangle - \langle w_A^{(t)}, \Psi(x,y^-)\rangle\big) \\
=&\langle w_A^{(t+1)} - w_A^{(t)}, \Psi(x,y)\rangle + \big(\langle w_A^{(t)}, \Psi(x,y^-)\rangle - \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_{i^-}^{(t+1)}, \Psi(x,y')\rangle\big) \\
\geq&\eta\big(\|\Psi(x,y)\|^2 - \langle \Psi(x,y), \Psi(x,y^-)\rangle\big) + 0 \\
=&\eta.
\end{aligned}$$

The last inequality holds because of (5) and (6), given the fact that the calibration operation only affects w_A's prediction on y and y^-. Hence (2) is proved in case I.

In case II, the MPMC-margin is

$$\rho_H^{(t)}(x,y) = \langle w_A^{(t)}, \Psi(x,y)\rangle - \langle w_B^{(t)}, \Psi(x,y^-)\rangle \leq 0. \tag{10}$$

Then the maximum value on y^- is predicted by w_B:

$$\langle w_B^{(t)}, \Psi(x,y^-)\rangle \geq \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_B^{(t)}, \Psi(x,y')\rangle, \tag{11}$$

$$\langle w_B^{(t)}, \Psi(x,y^-)\rangle \geq \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_A^{(t)}, \Psi(x,y')\rangle. \tag{12}$$

The calibration operation is:

$$\begin{aligned}
w_A^{(t+1)} &= w_A^{(t)} + \eta\Psi(x,y), \\
w_B^{(t+1)} &= w_B^{(t)} - \eta\Psi(x,y^-).
\end{aligned} \tag{13}$$

Similar to (8) and (9), it is easy to see $\langle w_A^{t+1}, \Psi(x,y)\rangle$ remains the maximum predictor on y at $t+1$. Then for either $i^- \in \{A, B\}$

$$\begin{aligned}
&\rho_H^{(t+1)}(x,y) - \rho_H^{(t)}(x,y) \\
=&\big(\langle w_A^{(t+1)}, \Psi(x,y)\rangle - \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_{i^-}^{(t+1)}, \Psi(x,y')\rangle\big) - \big(\langle w_A^{(t)}, \Psi(x,y)\rangle - \langle w_B^{(t)}, \Psi(x,y^-)\rangle\big) \\
=&\langle w_A^{(t+1)} - w_A^{(t)}, \Psi(x,y)\rangle + \big(\langle w_B^{(t)}, \Psi(x,y^-)\rangle - \max_{y' \in \mathcal{Y}\setminus\{y\}} \langle w_{i^-}^{(t+1)}, \Psi(x,y')\rangle\big) \\
\geq&\eta\|\Psi(x,y)\|^2 + 0 \\
=&\eta.
\end{aligned}$$

The second term is lower bounded by 0 because of (11) and (12), combined with the fact that the changes on w_A and w_B only affects the prediction on y and y^-. This completes the proof in case II. Since case III is symmetric to II, and IV is symmetric to I, we can conclude that the MPMC-margin on (x,y) increases by at least η after the calibration operation. □

Appendix B: Supplement to the experiments

We provide the code for reproducing toy example in main paper, and some implementation details for benchmark and multi-lingual experiments. Because the code of benchmark and multi-lingual experiments need to be run on specific deep learning environment (keras 2.2.0 [Chollet et al., 2015], tensorflow 1.9.0 [Abadi et al., 2015]), and a complete run on latter experiment requires ~10GB data over 20 hours on a single Titan Xp GPU. It is difficult to reproduce these experiments only with source code at review stage, and we will make them open-source after acceptance, including preprocessed data.

B.1 Code for reproducing the toy example

Run `jupyter notebook` in the command line and then open file `HMR_toy_example.ipynb` through your browser. Visualizations will be drawn on the page. Common Python 3 distributions (Anaconda, Python(x,y), ...) include all the required packages.

Please visit `https://github.com/YuriWu/HMR/` for updated toy example and other helpful codes.

B.2 Parameter setting for the benchmark experiment

We use the network structure in Google Colab [1]. Each local model is trained by Adam [Kingma and Ba, 2014] optimizer with learning rate 1e-4, 50 epochs, batch size 256.

B.3 Details for the multi-lingual experiment

The CNN structures used in our multi-lingual handwritting recoginition experiment are shown in Table B.1. The structures for Hiragana/Katakana/Kanji are the best structures used in Tsai [2016]. We tried some structures and find out good ones for the other scripts. The performance of HMR can be better if script-specific "tricks" are implemented into these local models. For example, The performance on Hangul can be improved about 1% by hybrid learning proposed in Kim and Xie [2015]. The single neural network in non-private setting over the entire combined dataset uses the same structure as Devanagari, which is the best we found among the four structures on the entire data.

Table B.1: ConvNet configurations for different scripts. The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

<table>
<tr><td colspan="4">ConvNet Configuration</td></tr>
<tr><td>Letters</td><td>Hiragana/Kanji</td><td>Devanagari/Hangul</td><td>Katakana</td></tr>
<tr><td>conv3-64
conv3-64</td><td>conv3-64</td><td>conv3-64
conv3-64</td><td>conv3-64
conv3-64</td></tr>
<tr><td colspan="4">maxpool</td></tr>
<tr><td>conv3-128
conv3-128</td><td>conv3-128</td><td>conv3-128
conv3-128</td><td>conv3-128
conv3-128</td></tr>
<tr><td colspan="4">maxpool</td></tr>
<tr><td>conv3-256
conv3-256</td><td>conv3-192</td><td>conv3-256
conv3-256</td><td>conv3-256
conv3-256</td></tr>
<tr><td colspan="4">maxpool</td></tr>
<tr><td rowspan="2"></td><td>conv3-256</td><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512
conv3-512</td></tr>
<tr><td colspan="3">maxpool</td></tr>
<tr><td>FC-512
FC-512</td><td>FC-1024
FC-1024</td><td>FC-1024
FC-1024</td><td>FC-4096
FC-4096</td></tr>
<tr><td colspan="4">FC-#classes</td></tr>
<tr><td colspan="4">softmax</td></tr>
</table>

We use Adam [Kingma and Ba, 2014] optimizer with learning rate 1e-4. The batch size is set to 128 and we train the models for at most 50 epochs if the multi-class cross-entropy loss does not converge earlier.

The reserved class output is implemented by creating a new neuron at the last layer, and use the average weights of other neurons at the same layer to initialize the weights of this new neuron. Each local model will be retrained with augmented data for one epoch at calibration operation.

[1] `https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/fashion_mnist.ipynb`

References

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015.

François Chollet et al. Keras. `https://keras.io`, 2015.

In-Jung Kim and Xiaohui Xie. Handwritten hangul recognition using deep convolutional neural networks. *IJDAR*, 18(1):1–13, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Charlie Tsai. Recognizing handwritten japanese characters using deep convolutional neural networks, 2016.