A Co-Adaptation in Subset-based FL

Federated Dropout is originally inspired by regular dropout [40], a regularization technique that constrains the capacity of a large NN model by randomly dropping parameters from the training, thereby limiting co-adaptation among parameters of the model. This is essential to improve the accuracy and reduce the over-fitting of parameters, as shown in various studies. FD and FedRolex adopt the dropout technique, removing CNN's filters in a round-based manner. These techniques, however, exercise dropout to its extreme, dropping a large part of filters so that not enough co-adaptation between filters remains. In particular, the gradients for the subset of parameters trained on a device are calculated without consideration of the error of the remaining parameters that reside on the server. These subsets are randomly changing over time and devices, reducing the co-adaptation of this distributed training process. Add to these the fact that the data is also distributed over devices, so applying such a random scheme significantly decreases the chance that a subset of parameters is being trained together over a sizable proportion of the data.

To further study the effects of co-adaptation on the reachable accuracy and the differences between FD and FedRolex, we run the following experiment, using CIFAR10 with ResNet20 and $s_{\text{FD/FedRolex}} = 0.25$:

- We modify FD s.t. all devices train the same random subset per round, i.e., the same indices $I^{(r,c)} = I^{(r)}$ per round (index k is omitted for simplicity).
- We limit the randomness by randomly selecting $I^{(r)}$ out of a set $\mathcal{I} = \{I_1, \ldots, I_{|\mathcal{I}|}\}$ of randomly initialized subsets that are generated once prior to training. Each round, $I^{(r)}$ is a random selection out of $\mathcal{I}$.

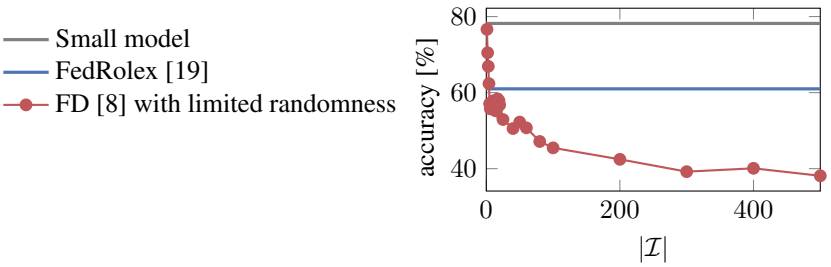

Figure 5: FD with limited randomness using CIFAR10 with ResNet20 and $s_{\text{FD/FedRolex}} = 0.25$.

We vary the randomness by varying the number of sampled subsets in $\mathcal{I}$, s.t. $|\mathcal{I}| \in [0, 500]$. Thereby, the probability of a specific subset being selected is $p = \frac{1}{|\mathcal{I}|}$. Therefore, all the devices train the same submodel in a round, and there are only $|\mathcal{I}|$ submodels that would be trained by devices over the training period. Evaluation is done with the full server model.

We observe the following effects: Firstly, the final accuracy drops proportionally to $\sim p$. Secondly, it can be observed that in the case of $|\mathcal{I}| = 1$, FD behaves similarly to the small model baseline, as always the same subset is used for training. We also observe that remaining untrained filters have a minor effect on the accuracy when compared with a small model. However, because of these untrained parameters, the model fails to reach higher accuracies as with SLT (see Section 3). The accuracy drops with introducing more randomness to the training process (i.e., increasing $|\mathcal{I}|$). Lastly, it can be observed that the rolling window approach of FedRolex is a special case of FD with limited randomness (i.e., $|\mathcal{I}| = 5$ in this experiment).

B Training Memory Measurements in PyTorch

We measure the maximum memory requirement for the evaluated NN models ResNet and DenseNet using PyTorch 1.10. Specifically, we measure the size of the activations, gradients, and weights. These memory measurements are done offline (prior to training) and do not require any data.

- **Measurement of weights**: To measure the size of the weights, we sum up all tensors that are present in the NN's `state_dict`.

- **Measurement of activations and gradients**: To measure the size of the activations that have to be kept in memory, as well as the gradients, we apply `backward_hooks` to all relevant PyTorch modules in an NN. Specifically, we add these hooks to `Conv2d`, `BatchNorm2d`, `ReLU`, `Linear`, and `Add` operations. If a hook attached to a module is called, we add the respective size of the activation map and the size of the calculated gradient to a global variable add up all required activations and gradients.