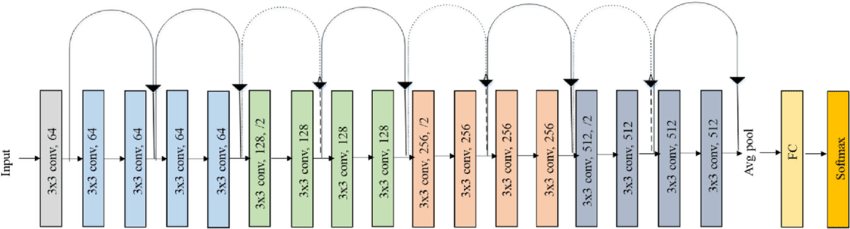
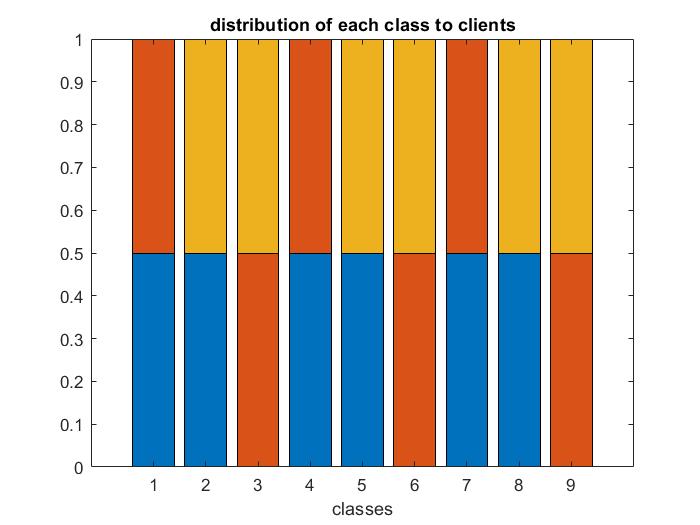
In the last chapter we have shown, through federated learning, clients can learn recognizing scene categories that they have no samples of. Expanding on that we want to know whether we can learn not only different scene categories like office, pantry etc. but also different places among those categories. For example, can a client learn to differentiate between different offices some of which it has never seen before?

First, we had to decide how our dataset should be. Given the fact that recognizing inner-scene differences is a more challenging task than recognizing differences between scenes, we need a more robust feature extractor and higher quality images. We decided to use ResNet-18 -which is much deeper than the CNN architecture we used in the last chapter- accompanied by 160x160 images. When we look at the original Resnet-18 paper(<https://arxiv.org/abs/1512.03385>) we see that they have used 224x224 images and those images went through 5 halving in total -reduced to 1/32th of original size in each side- till they reach the average pooling layer as 7x7 channels. We believed using 224x224 images might be too computationally expensive, so we used 160x160 images which meant we had 5x5 channels at the average pooling stage. All this Resnet-18 architecture until the Fully Connected layer at the very end is called the feature extractor.

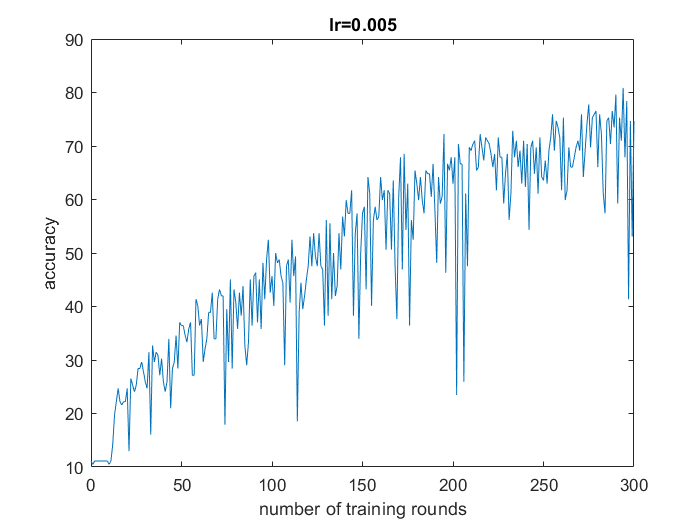
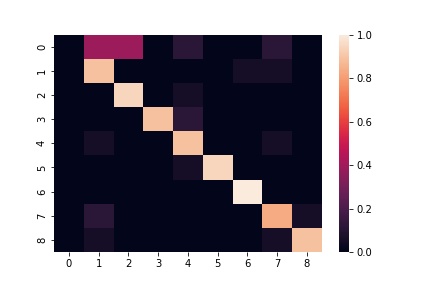


Now that we had determined our model and the image sizes, we now must determine the size of the dataset. We decided to use 3 scene categories (office, lobby, hallway) with 3 places from each scene category so in total we have 9 classes/places. Places 0,1 and 2 are offices; 3,4, and 5 are lobbies; 6,7, and 8 are hallways. We picked 90 images randomly from each place and used 20% of those as test set. It should be noted that we didn’t pick those places randomly instead we made sure 90 images from a place were enough to have a good representation of that place. So, the places we took those 90 pictures from had around 100 to 200 images in total. This constraint excluded places from scenes like auditorium which had more than 2000 images each. It would be impossible to pick 90 images that could represent the feature space of such places adequately. Before moving on to federated learning we checked the model in a central machine learning setting to make sure the model-image properties are sufficient. At the end of this very standard machine learning training, we got the [central confusion matrix] with X overall accuracy.

kare içeren bir resim

Açıklama otomatik olarak oluşturuldu 

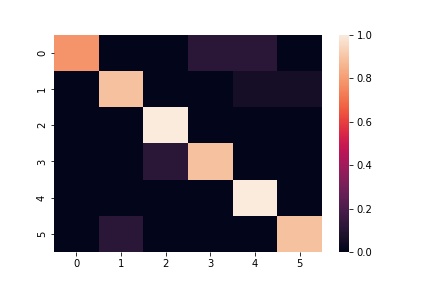
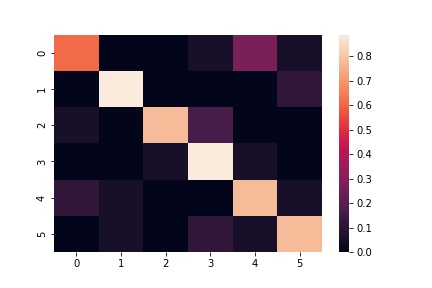
As we can see from the confusion matrix all the places except the first one separated very well and the samples from first one were sometimes predicted as yet another place of same scene category -office. It should be noted further accuracy improvements can be observed by training longer, but our goal was not to get the best central training result rather show training does work. We concluded the Resnet-18 using 160x160 images is robust enough to learn multiple places from multiple scenes. Moving on to the federated setting class distributions to clients can be found at [distribution of each class to clients]. Here we have used 3 clients in total and this number was decided due to computational constraints. Each client has samples from 6 out of 9 places and each place is available at the dataset of 2 out of 3 clients. It is important to note while 2 different clients may have training samples from same place, the samples are not same/repeated; each of those two clients hold half of the samples of that place. Using the dataset and the distribution we have described, we run the federated averaging for 300 rounds with each client participating to each communication round.

Looking at the accuracy plot, we can see our system is not very stable despite the fact accuracy kept improved overall. Looking at the last chapter we see that the simpler CNN architecture was much more stable. We speculate as the architecture gets deeper and deeper small differences can have larger overall effects and can cause dips in accuracy. This instability led us to use 0.005 learning rate for client optimizers (SGD). Beforehand we were using 0.01 but when we run our trial runs with that learning rate the instability was even worse and it was hindering accuracy improvements. Using a smaller learning rate enabled us to achieve somewhat steady accuracy improvement. At the end of the 300 rounds, we have achieved 76% accuracy. Looking at the confusion matrix we see that all the places except the first one has separated very nicely. First one was almost always predicted as second and third places which are from the same scene as the first one -offices. This is a very promising result since it implies the model has learned features of a place are most similar to the features of places from the same scene. Also, we should remember the first place was already the worst performing one in central training.

An open problem in federated learning is generalization to novel clients (<https://arxiv.org/pdf/1912.04977.pdf>). Taking inspiration from transfer learning and personalized federated learning via layer sharing (<https://arxiv.org/pdf/1912.00818.pdf>) we decided to check if the feature extractor we got by federated learning is capable of extracting the features of novel places well enough to distinguish between them. First, let’s decide how our dataset should be? As we stated the goal is measuring the generalizability of our model to novel instances, so it makes sense to choose places from scenes that were not present in our dataset. Additionally, we would like to pick at least 2 places from each novel scene to investigate how well we can differentiate between places from the same scene. Keeping those in mind we decided to use 3 scenes (conference room, lounge, and storage) and pick 2 places from each one those scenes. So, in total we will have 6 novel places in our new dataset. We once again decided to pick 90 images from those places with test ratio, 0.2, and image size, 160x160, is kept same. While picking places from scenes we kept in mind the constraint we mentioned earlier: if a place has too many images it isn’t likely to pick 90 images that represents this place well enough.

Now that we finalized our new dataset, let’s decide how exactly training will work. From our transfer learning experience and the intuition we derived from layer sharing paper, the first idea we got is to freeze the feature extractor part of our model (from beginning to average pooling layer included), plug in a new fully connected layer with 6 outputs and train it. Here it should be noted, since the feature extractor part of the model is frozen, throughout the training we will be getting same features as inputs of fully connected layer. Since only this fully connected layer is being trained, we could in theory extract the features of all the images and simply train a fully connected layer with those features as its inputs. We have done very similar exercises to that while taking EE573. The accuracy we got by doing that was roughly 82%. The confusion matrix of that can bee seen at [alt soldaki]

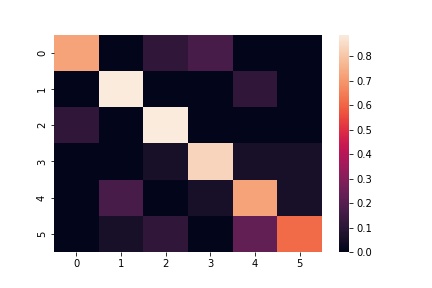
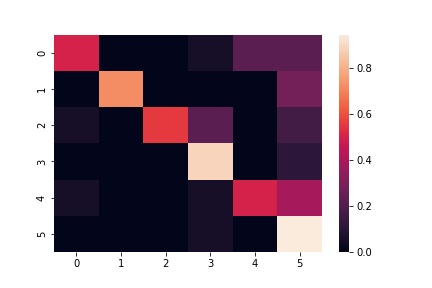


As we explained the features we extract from images by using the frozen feature extractor can also be considered as the inputs. For that reason, what we had done by training a Resnet-18 with frozen feature extractor is same as training a single layer of fully connected layer with only inputs changing: images themselves for the former and the features of those images for the latter. This is an important point since there aren’t many techniques to train with image data yet there are many techniques when it comes to a training with a list of features. One such technique that we covered in EE573 was Linear Discriminant Analysis (LDA) and we decided to apply that technique to our task at hand. To do that we essentially followed the feature extraction we have just outlined. We extracted the features of all the data, fit our LDA classifier to training samples and then tested for the test set. We got 91% accuracy with LDA. The confusion matrix can be seen at [üstteki sağda]. Since LDA work by finding axes that project good class separation, getting a good accuracy with LDA is a great sign of our feature extractor’s capacity to extract distinct features from different scenes and places.

The fact that LDA was the higher achiever of those two methods might seem a bit off but we should keep in mind since we frozen the feature extractor both LDA and the Fully Connected layer were getting the same inputs. So we only compared the stand alone performance of LDA and a single Fully Connected layer. So why did LDA outperformed the Fully Connected layer? After all they both are linear (We can only get nonlinearity in Deep Neural Networks by using nonlinear activation functions like ReLU between neural layers. The layers themselves and their combinations would otherwise always be linear.) We speculate accuracy discrepancy might indicate performance improvements with further training with smaller/decaying learning rate or maybe stacked fully connected layers might lead to improvements but doing so would require using a different training scheme or model architecture, respectively.

One more thing we have noticed is in the previous test when the model predicted a wrong place it was very likely it still predicted a place from the scene the label place belonged to. Like how it was predicting the wrong office but still an office. But now we see a lot of predictions that were not only the wrong place but also the wrong scene. We speculate this performance degradation is because our model was not trained and hence optimized for the feature space of these scenes. So, we are still using the features from previous scenes and when new features from different scenes resemble the same set of features from previous scenes the predictions might be off.

We have shown when we train a feature extractor with federated learning, we can use that feature extractor as frozen backbone of other client models that has completely different scenes. It is intuitive we will have better performance with such clients if we train our feature extractor with a more diverse dataset since that enables learning more features. We wanted to show that is indeed the case. To achieve that we have deployed a new federated learning scheme that’s nearly same as the previous one with the one exception: we removed the hallway places from the dataset. So, our dataset now consists of three offices and three lobbies. We have trained our model in federated setting with the exact same parameters using this new dataset. We have achieved 82% accuracy in our test set which is an improvement over the 76% we got last time. Yet last time we were classifying 9 places and we are now classifying 6. So, this improvement shall be considered with that in mind.



After we finished training the feature extractor, we followed the same previous steps to train a Fully Connected layer and LDA. Confusion matrixes we got can be seen at [sol fc, sağ lda]. The accuracy of Fully Connected layer was 62% and LDA got 77%, so LDA outperformed FC once again. As we predicted the accuracy fell when we used a feature extractor that was trained with less diverse dataset. One other observation is just how frequently we predicted the wrong scene. We again speculate this is because our feature extractor only knows features from 6 places and 2 scenes it was trained with. The FC layer and LDA is trying to assign patterns of these features to the new scenes, but they aren’t quite successful due to poor quality of feature extractor. So we conclude the more diverse training data we use for feature extractor, the more robust it becomes adapting at novel scenes.