5043 Advanced Machine Learning - HW 4

Author: Enzo Durel

April 1, 2025

Contents

List of Figures

1 Figures

1.1 Figure 1a

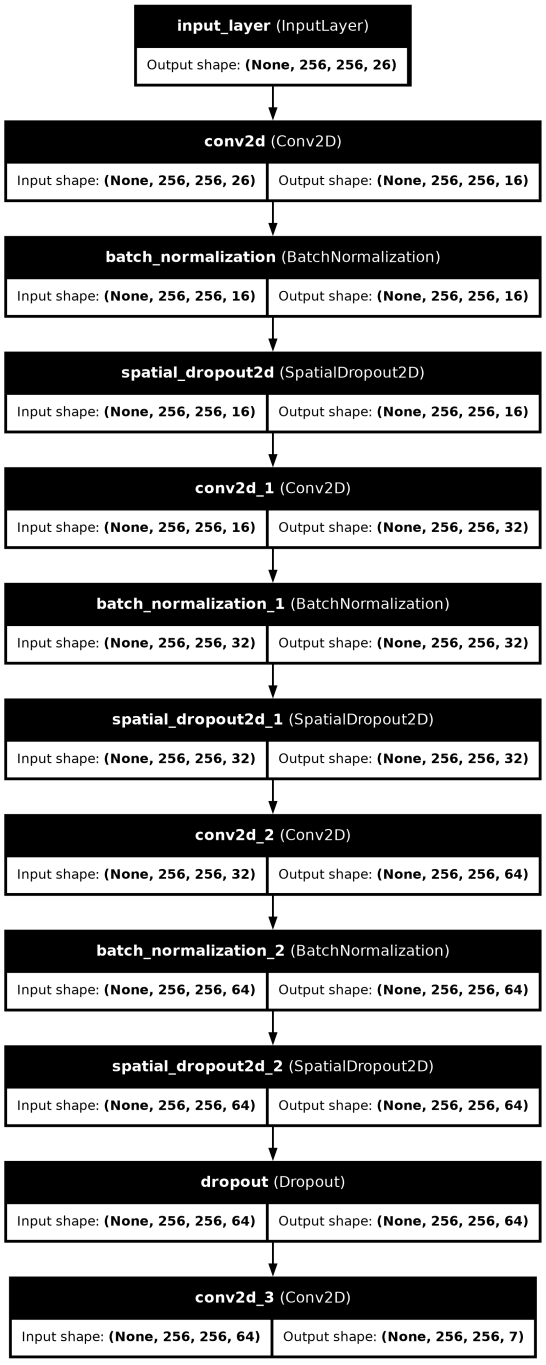
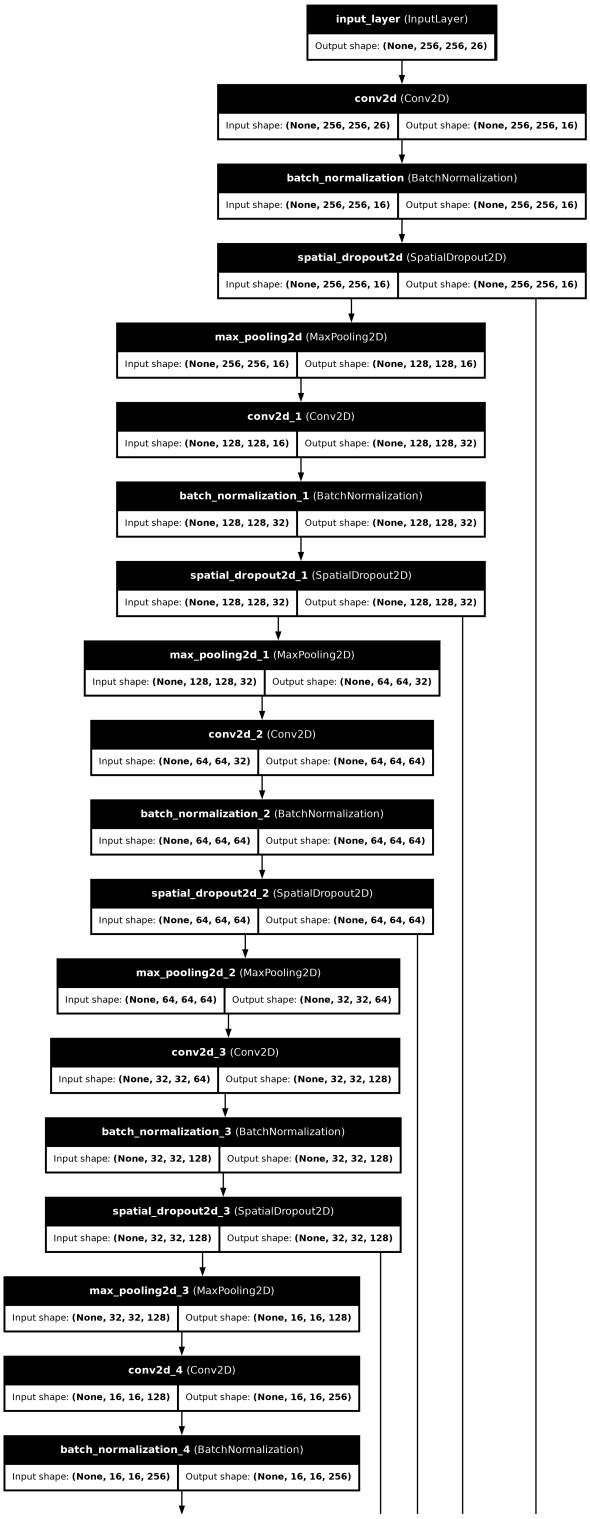
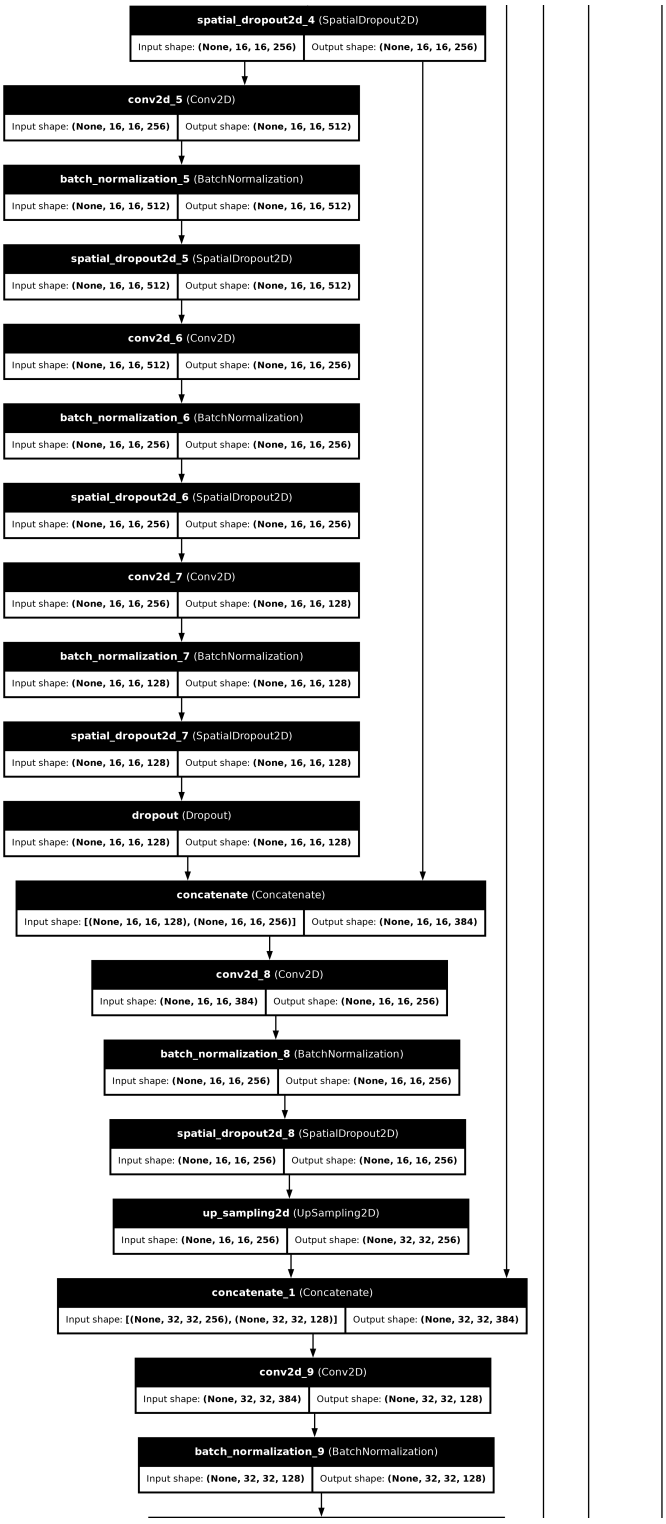


Fig. 1: Shallow Model Architecture

1.2 Figure 1b


input_layer (InputLayer)
Output shape: (None, 256, 256, 26)
conv2d (Conv2D)
Input shape: (None, 256, 256, 26)
Output shape: (None, 256, 256, 16)
batch_normalization (BatchNormalization)
Input shape: (None, 256, 256, 16)
Output shape: (None, 256, 256, 16)
spatial_dropout2d (SpatialDropout2D)
Input shape: (None, 256, 256, 16)
Output shape: (None, 256, 256, 16)
max_pooling2d (MaxPooling2D)
Input shape: (None, 256, 256, 16)
Output shape: (None, 128, 128, 16)
conv2d_1 (Conv2D)
Input shape: (None, 128, 128, 16)
Output shape: (None, 128, 128, 32)
batch_normalization_1 (BatchNormalization)
Input shape: (None, 128, 128, 32)
Output shape: (None, 128, 128, 32)
spatial_dropout2d_1 (SpatialDropout2D)
Input shape: (None, 128, 128, 32)
Output shape: (None, 128, 128, 32)
max_pooling2d_1 (MaxPooling2D)
Input shape: (None, 128, 128, 32)
Output shape: (None, 64, 64, 32)
conv2d_2 (Conv2D)
Input shape: (None, 64, 64, 32)
Output shape: (None, 64, 64, 64)
batch_normalization_2 (BatchNormalization)
Input shape: (None, 64, 64, 64)
Output shape: (None, 64, 64, 64)
spatial_dropout2d_2 (SpatialDropout2D)
Input shape: (None, 64, 64, 64)
Output shape: (None, 64, 64, 64)
max_pooling2d_2 (MaxPooling2D)
Input shape: (None, 64, 64, 64)
Output shape: (None, 32, 32, 64)
conv2d_3 (Conv2D)
Input shape: (None, 32, 32, 64)
Output shape: (None, 32, 32, 128)
batch_normalization_3 (BatchNormalization)
Input shape: (None, 32, 32, 128)
Output shape: (None, 32, 32, 128)
spatial_dropout2d_3 (SpatialDropout2D)
Input shape: (None, 32, 32, 128)
Output shape: (None, 32, 32, 128)
max_pooling2d_3 (MaxPooling2D)
Input shape: (None, 32, 32, 128)
Output shape: (None, 16, 16, 128)
conv2d_4 (Conv2D)
Input shape: (None, 16, 16, 128)
Output shape: (None, 16, 16, 256)
batch_normalization_4 (BatchNormalization)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)

1.2 Figure 1b


spatial_dropout2d_4 (SpatialDropout2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)
conv2d_5 (Conv2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 512)
batch_normalization_5 (BatchNormalization)
Input shape: (None, 16, 16, 512)
Output shape: (None, 16, 16, 512)
spatial_dropout2d_5 (SpatialDropout2D)
Input shape: (None, 16, 16, 512)
Output shape: (None, 16, 16, 512)
conv2d_6 (Conv2D)
Input shape: (None, 16, 16, 512)
Output shape: (None, 16, 16, 256)
batch_normalization_6 (BatchNormalization)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)
spatial_dropout2d_6 (SpatialDropout2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)
conv2d_7 (Conv2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 128)
batch_normalization_7 (BatchNormalization)
Input shape: (None, 16, 16, 128)
Output shape: (None, 16, 16, 128)
spatial_dropout2d_7 (SpatialDropout2D)
Input shape: (None, 16, 16, 128)
Output shape: (None, 16, 16, 128)
dropout (Dropout)
Input shape: (None, 16, 16, 128)
Output shape: (None, 16, 16, 128)
concatenate (Concatenate)
Input shape: [(None, 16, 16, 128), (None, 16, 16, 256)]
Output shape: (None, 16, 16, 384)
conv2d_8 (Conv2D)
Input shape: (None, 16, 16, 384)
Output shape: (None, 16, 16, 256)
batch_normalization_8 (BatchNormalization)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)
spatial_dropout2d_8 (SpatialDropout2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 16, 16, 256)
up_sampling2d (UpSampling2D)
Input shape: (None, 16, 16, 256)
Output shape: (None, 32, 32, 256)
concatenate_1 (Concatenate)
Input shape: [(None, 32, 32, 256), (None, 32, 32, 128)]
Output shape: (None, 32, 32, 384)
conv2d_9 (Conv2D)
Input shape: (None, 32, 32, 384)
Output shape: (None, 32, 32, 128)
batch_normalization_9 (BatchNormalization)
Input shape: (None, 32, 32, 128)
Output shape: (None, 32, 32, 128)

1.2 Figure 1b

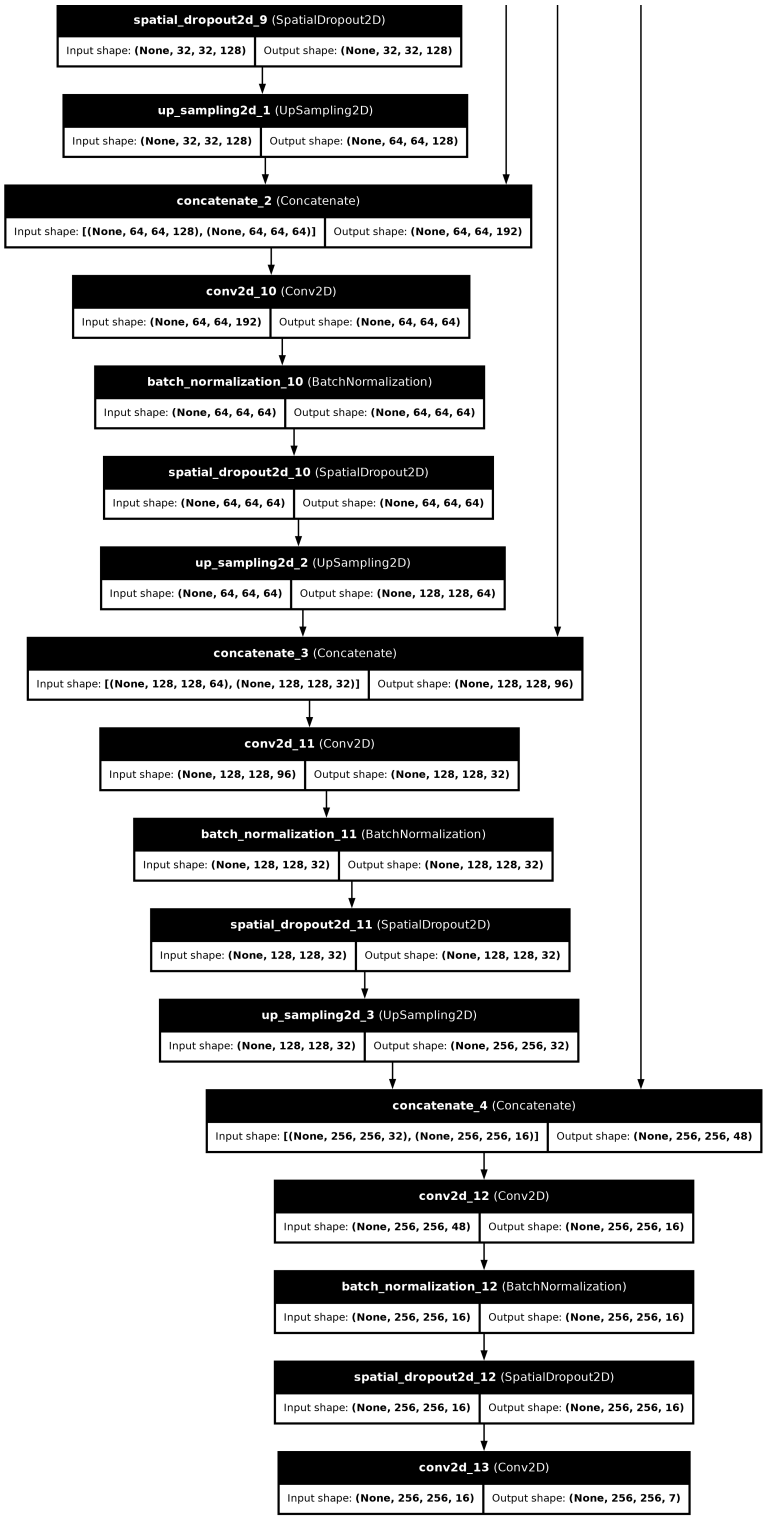


Fig. 2: Deep Model Architecture

We can see in the deep model architecture the "skip lines" used by the U-Net. We can also see the encoder/decoder pattern clearly.

1.3 Figure 2a

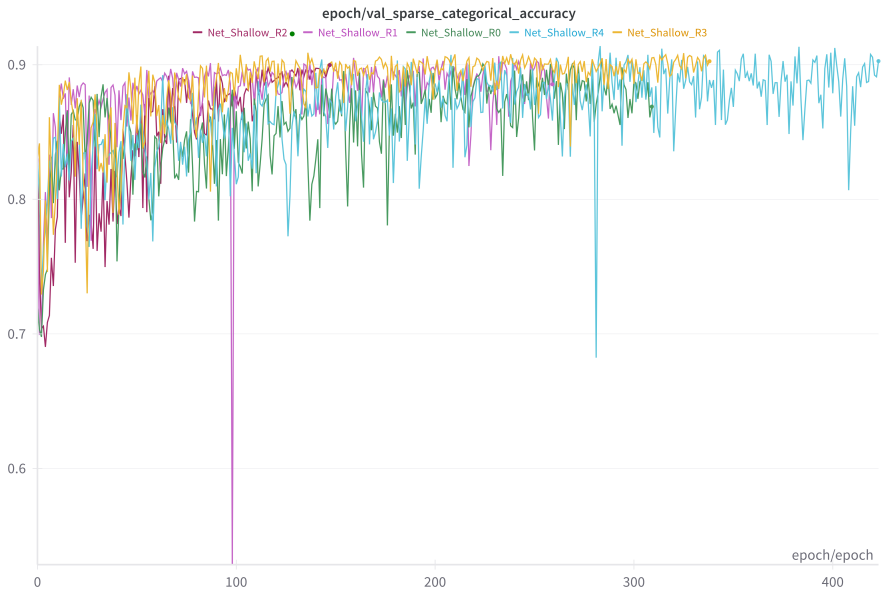


Fig. 3: Validation Accuracy as a function of epoch for the Shallow model

1.4 Figure 2b

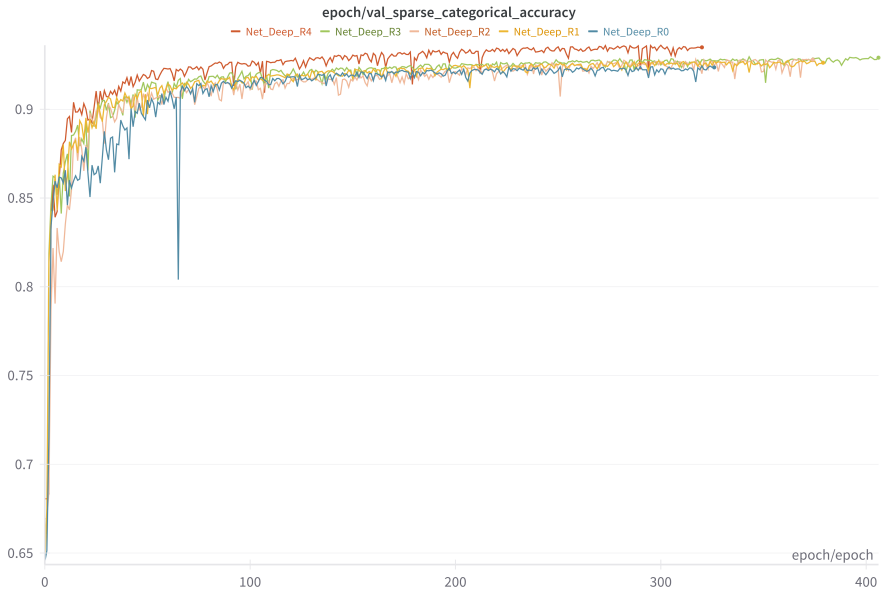


Fig. 4: Validation Loss as a function of epoch for the Deep models

We can see that deep models are a lot more stable than shallow models during the training.

1.5 Figure 3a

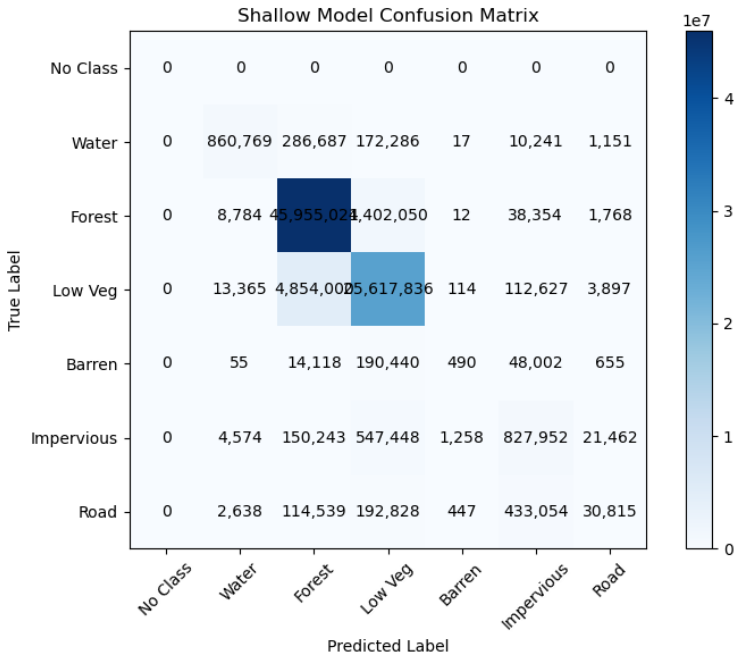


Fig. 5: Confusion Matrix of the test set data across all rotations for the Shallow model

1.6 Figure 3b

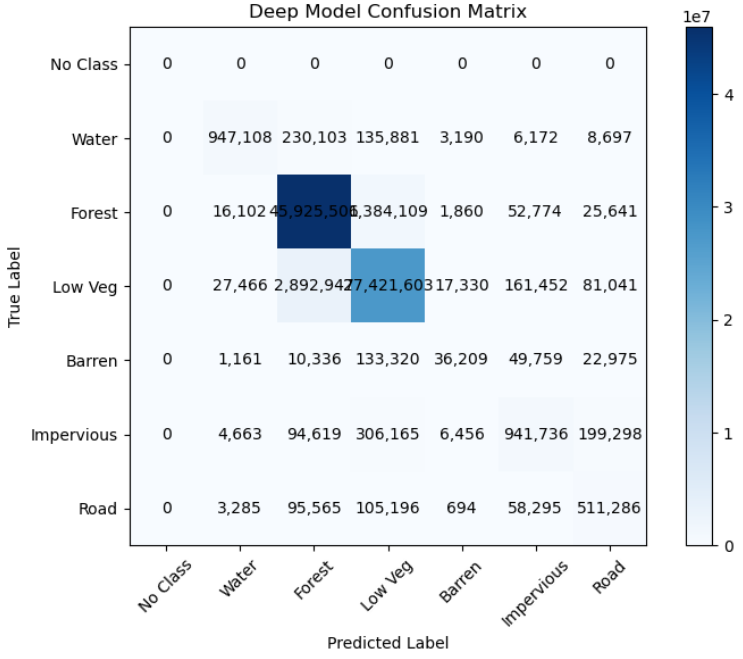


Fig. 6: Confusion Matrix of the test set data across all rotations for the Deep model

We can see that the deep model correctly detects more of all classes but also makes more "smoothly" the misdetection in other classes. It seems that there are a lot of class 2 and 3 and it is the main difficulty to clearly identify these without mixing them up.

1.7 Figure 4

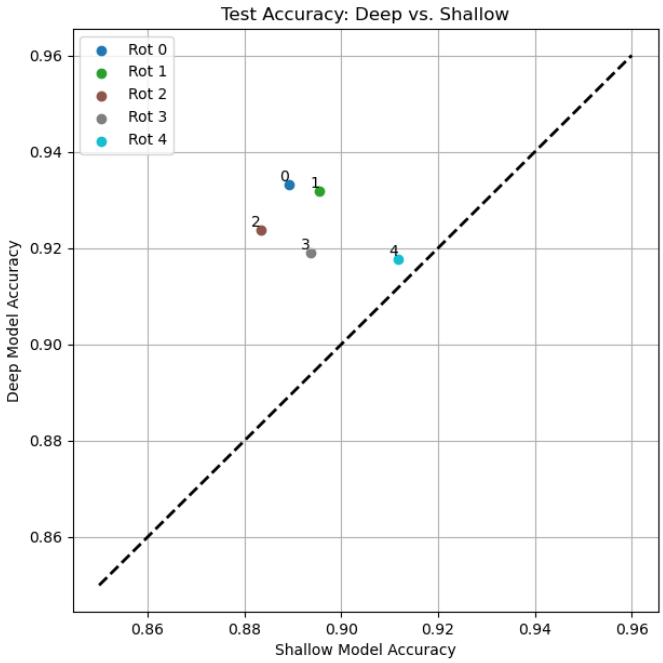


Fig. 7: Test set accuracy for the deep vs shallow networks

We clearly see that the deep model is more accurate in every rotation (folds).

1.8 Figure 5a

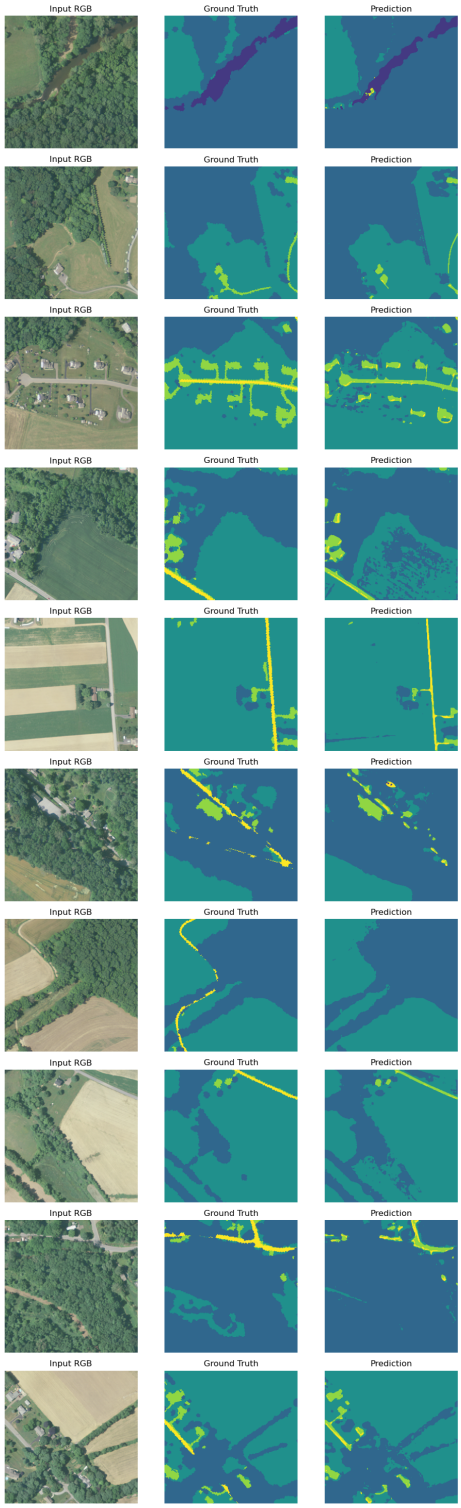


Fig. 8: Example Predictions for Shallow Model

1.9 Figure 5b

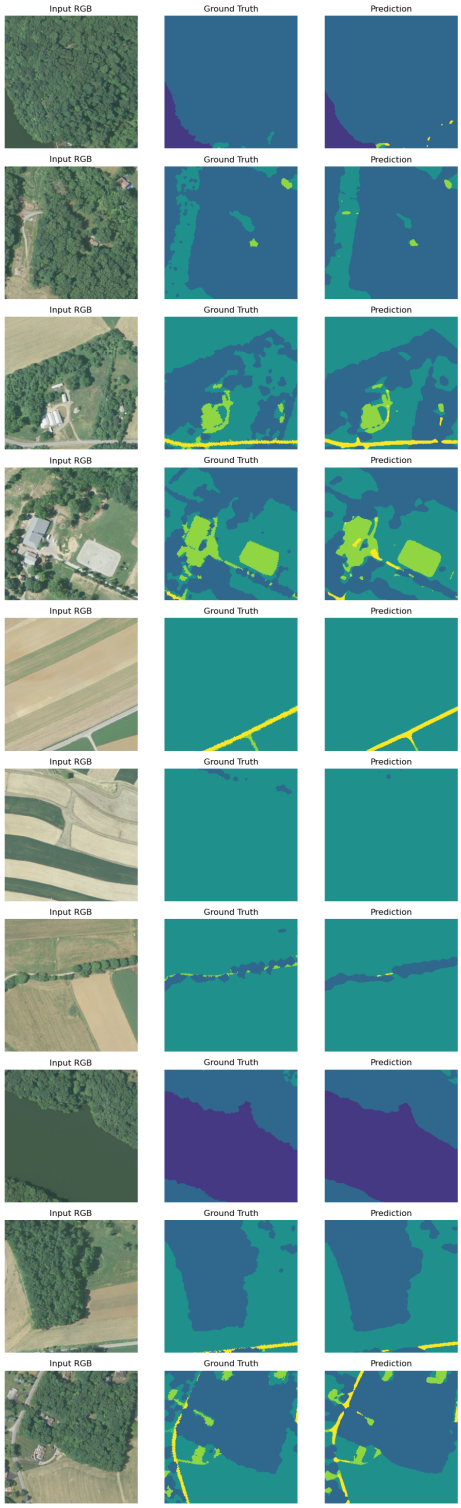


Fig. 9: Example Predictions for Deep Model

We can see that the shallow model struggles to identify big areas and makes prediction very "noisy". The deep model makes "smoother" prediction and sometimes makes it too smooth and misses some key points.

2 Analysis & Discussion

2.1 *"What regularization choices did you make for your shallow and deep networks? Why?"*

I chose to use l2 regularization, dropout and spatial dropout for my both models. I have a l2 equal to 0.0001, a dropout of 0.2 and a spatial dropout of 0.1 for the shallow model. I have a l2 equals to 0.0001, a dropout of 0.4 and a spatial dropout of 0.2.

The regularization is bigger in my deep model because bigger models tend to overfit more than shallow ones. For my deep model I also add batch normalization to accelerate training and add some "noise" which can help not to overfit.

2.2 *"How do the training times compare between the two model types?"*

The training times for my shallow model takes on average 45 minutes. The training of the deep model takes me around 1h and 10 minutes. I have a learning rate for my shallow network bigger than the one on the deep model. The goal here was to avoid overfitting because the deep model is bigger than the shallow model.

We can see that the deep model performs better than the shallow model (88%-90% vs. 92%-93%). Moreover, we can see that the validations curves are more stable for deep models than shallow models.

Finally, the training time of the shallow model is faster than the deep model training time.

2.3 *"Describe the relative test set performance of the two model types."*

2.3.1 Shallow Network Accuracy Results

Table 1: Shallow Model Sparse Categorical Accuracy Accross 5 Folds

	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Min	Max	Mean
Sparse Categorical Accuracy	0.88336	0.8893	0.91178	0.89544	0.89362	0.88336	0.91178	0.8947

2.3.2 Deep Network Accuracy Results

Table 2: Deep Model Sparse Categorical Accuracy Accross 5 Folds

	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Min	Max	Mean
Sparse Categorical Accuracy	0.9191	0.91768	0.9332	0.9318	0.92368	0.91768	0.9332	0.925092

We can see that the deep model have better results than the shallow model, we can see that with the mean, min and max accuracies showed in the tables. We can also see this in the scatter plot figure where the point are above (on the side of deep model) the diagonale.

2.4 "Describe any qualitative differences between the outputs of the two model types. What types of errors do your models tend to make?"

2.4 *"Describe any qualitative differences between the outputs of the two model types. What types of errors do your models tend to make?"*

We can see that the deep network has better performance with more details compared to the shallow network which can ignore entire zones. Moreover, we can see that the deep model correctly detects more of all classes but also makes more "smoothly" the misdetection in other classes. It seems that there are a lot of class 2 and 3 and it is the main difficulty to clearly identify these without mixing them up. This is clearly demonstrate in the predicted examples images.

This makes the deep model more reliable than the shallow network and show that a little difference between accuracies can make a big deal in terms of quality representation.