Supplement of Biogeosciences, 22, 6509–6543, 2025
https://doi.org/10.5194/bg-22-6509-2025-supplement

Biogeosciences Open Access EGU

Supplement of

Identifying alpine treeline species using high-resolution WorldView-3 multispectral imagery and convolutional neural networks

Laurel A. Sindewald et al.

Correspondence to: Laurel A. Sindewald (laurel.sindewald@ucdenver.edu)

S1 Convolutional Neural Network (CNN) Methods

CNNs are a form of deep learning model that are useful for gridded data. The models are spatially aware and can detect patterns in images or other gridded data on multiple scales. CNNs are a popular method for classification of remote sensing imagery. A more detailed overview of CNNs can be found in section 2.5 of the methods section in the main manuscript.

S1.1 How the CNN Learns through Backpropagation

Before training, the weights and biases are randomly initialized. As the CNN learns, weights are adjusted through a process called gradient descent with backpropagation (Goodfellow and Bengio and Courville, 2016a). The end goal is to minimize the model's error, estimated with a loss function. (Often neural networks are trained with maximum likelihood, using the negative log-likelihood or cross-entropy between the training and observation distributions (Goodfellow and Bengio and Courville, 2016a). See Appendix B for a discussion of specific loss functions.) For each weight w, the adjustment is $-\alpha \frac{\partial J}{\partial w}$, where α is the learning rate; J is the loss; and $\frac{\partial J}{\partial w}$ is the partial derivative of the loss with respect to w. The learning rate must be positive and is typically a small number ($\ll 1$). This adjustment is called "gradient descent," as it adjusts w in the direction opposite the loss gradient with respect to w. If w is in the output (deepest) layer, $\frac{\partial J}{\partial w}$ can be computed directly. However, if w is in a shallower layer $\mathcal{L}$, then $\frac{\partial J}{\partial w}$ cannot be computed directly, because the dependence of J on w is mediated by the weights in all layers between $\mathcal{L}$ and the output. Thus, for weights not in the output layer, one must apply the chain rule for differentiation. So, one must calculate the partial derivative of the loss with respect to each weight w_{deep} in the deepest layer, the partial derivative of each w_{deep} with respect to each weight in the second-deepest layer, and so on, until reaching layer $\mathcal{L}$. All of these partial derivatives are multiplied together to calculate the contribution of w in layer $\mathcal{L}$ to the loss. So, backpropagation is a combination of gradient descent with the chain rule, where weights and biases are learned simultaneously across all convolutional and dense layers. This process is repeated for every data sample in a training batch, and the changes to the weights and biases—determined by gradient descent with backpropagation—are averaged across all of the data samples.

Each partial derivative value may be very small, so if the network is deep enough, requiring the multiplication of many small numbers, the contribution of a weight to the overall loss can become so small ("vanishingly" small) that the computer cannot record it (e.g., x^{-30}) (Dubey and Jain, 2019). This is known as the vanishing gradient problem. If the computer cannot calculate and store the relative contribution of each weight to the loss, it does not have a gradient with which it could apply the gradient descent rule, and so it is not able to determine how the weights must be adjusted. The vanishing gradient problem was thus an early, major barrier to the construction of deeper networks. Fortunately, ML researchers have found

strategies to solve the vanishing gradient problem since the original development of CNNs in the 1980s (Fukushima and Miyake, 1982), including batch normalization and the rectified linear unit (ReLU) activation function (described in detail in section S1.3). Advances in computer processing and memory technology were also, of course, instrumental in enabling wider use of CNNs and other deep learning methods (Krizhevsky and Sutskever and Hinton, 2017).

S1.2 The Importance of Activation Functions and Batch Normalization

Each convolutional filter performs a linear operation (specifically the sum of $K_w * K_h * C_{in}$ products plus a bias value), and any series of linear operations is itself linear. Thus, a CNN with only convolutional layers could learn only linear relationships (Lagerquist, 2020). Activation functions introduce an elementwise non-linear transform after convolution, allowing CNNs to learn non-linear relationships (Dubey and Jain, 2019; Goodfellow and Bengio and Courville, 2016a). This function does not need to be in a specific form, as long as the CNN has some process by which it can learn non-linear relationships (Lagerquist and Mcgovern and Gagne Ii, 2019), so the activation function is typically simple for computational efficiency. Many activation functions, such as the sigmoid and hyperbolic tangent (tanh), confine the output to a range of values, such as [0, 1] or [-1, 1], respectively. If the values are squashed close to 0, however, this can contribute to the vanishing gradient problem (Maas, 2013; Dubey and Jain, 2019). Thus, the NN is more likely to get "stuck" in a suboptimal local minimum, rather than finding the global minimum.[1] In the early 2000s, rectifier nonlinearities were found to outperform sigmoid and tanh activation functions and to improve with model depth, without encountering vanishing gradients (Glorot and Bordes and Bengio, 2011; Maas, 2013).

[1]These are minima in a W-dimensional space, where W is the total number of weights + biases in the NN.

Now a commonly used activation function, the rectified linear unit (ReLU) preserves positive feature values but reduces the magnitude of negative values. Strict ReLU is defined as x' = max(0, x), where x is the input element and x' is the output. Thus, strict ReLU removes negative feature values completely by zeroing them out. Leaky ReLU, on the other hand, is defined as x' = max(α * x, x), where α is the slope parameter, ranging from (0, 1). Thus, leaky ReLU reduces the magnitude of negative feature values without zeroing them completely (Nair and Hinton, 2010). Importantly, ReLU is still a differentiable function, allowing for gradient-based optimization through backpropagation (Dubey and Jain, 2019; Goodfellow and Bengio and Courville, 2016a). ReLU reduces the risk of a vanishing gradient by preserving the magnitude of the positive values; as long as the ReLU is activated above 0, its partial derivative is 1 and it doesn't contribute to a vanishingly small gradient during the multiplication of many partial derivatives (Maas, 2013). The Strict ReLU activation function sets negative values to exactly 0 and was appealing early on because of its similarity to biological neural activation functions (Glorot and Bordes and Bengio, 2011; Goodfellow and Bengio and Courville, 2016a). (Early NN developers were greatly inspired by the neuroscience literature, though the models we have today are not thought to be direct approximations of biological neural networks (Goodfellow and Bengio and Courville, 2016a).) Unfortunately, neurons with weights set to 0 by strict ReLU can cease to be updated and become "dead neurons", where no learning occurs (Maas, 2013). Leaky ReLU solves the problem of dead neurons while preserving the performance advantages of ReLU (Maas, 2013; Dubey and Jain, 2019).

To further reduce the possibility of vanishing gradients, one can use batch normalization. During batch normalization, feature map elements are transformed to z-scores from a standard normal distribution, with mean of 0.0 and standard deviation of 1.0 (Lagerquist, 2020). Normalization helps prevent vanishing gradients by counteracting internal covariate shift toward saturation values at the limits/asymptotes of activation functions that can happen with the use of sigmoid and tanh activation functions (Ioffe and Szegedy, 2015). By eliminating the vanishing gradient problem—preventing the model from getting stuck in local minima—batch normalization allows for higher learning rates and increases the speed of model convergence (Ioffe and Szegedy, 2015). Batch normalization also puts all features in a map on the same scale, preventing the over-emphasis of features due solely to their original scale (e.g., elevation in meters vs. radiance in W m^{-2} sr^{-1} μm^{-1}). Lastly, batch normalization can act as a form of regularization; by reparameterizing the model, it introduces noise to the weights and biases (Goodfellow and Bengio and Courville, 2016b).

S1.3 Regularization Methods: How to Prevent Overfitting

With thousands of parameters in each convolutional and dense layer, CNNs quickly run the risk of overfitting the model to the data, preventing the model from generalizing well to new data (Goodfellow and Bengio and Courville, 2016c). To reduce this problem, ML developers use regularization methods, which encourage a simpler model by penalizing the model for being complex or by rewarding simplicity. The core idea of regularization is to improve the model's generalizability and parsimony without reducing its training error (Goodfellow and Bengio and Courville, 2016a). Three forms of regularization

used in the present work are the addition of penalties to the loss function (commonly L_1 and L_2), data augmentation, and dropout. Note that all three regularization methods are used only during training. At inference time (when applying a trained model to out-of-bag or validation data), all regularization methods are turned off—including, importantly, dropout and data augmentation.

The L_1 and L_2 penalties are known as the lasso and ridge penalties, respectively (Goodfellow and Bengio and Courville, 2016b). These penalties are added to the loss function (thus increasing the loss). The L_1 penalty, shown in Eq. S1, is the sum of absolute coefficient values, it is called the lasso penalty because it pushes model parameters to become exactly 0, thus reducing the effective number of parameters and drawing a "lasso" around the non-zero parameters that are kept.

$$L_1 = \sum_{j=1}^{M} |B_j| \quad \text{S1}$$

where M is the total number of parameters in the convolutional or dense layer, and B_j is the value of the jth parameter. The L_2 penalty, shown in Eq. S2, is the sum of the squared parameters, and pushes the weights to become small but not exactly zero.

$$L_2 = \sum_{j=1}^{M} {B_j}^2 \quad \text{S2}$$

Data augmentation is another method of regularization aimed to prevent overfitting (Krizhevsky and Sutskever and Hinton, 2017). Data augmentation is the practice of artificially inflating the dataset by adding small amounts of noise to the data to generate multiple samples from each original sample. Data augmentation changes only the predictor values; the changes are assumed to be small enough that they do not change the target value (in our case, the species label). The fake data samples are noisy enough to force the NN to apply real learning to recognize the signal through added variability, but not so noisy as to change the correct classification (Goodfellow and Bengio and Courville, 2016b). It is important to emphasize that data augmentation does not fix issues introduced by having a non-representative dataset; as with any type of statistical model or representation, the model is only generalizable to the limits of the original sampling framework. However, the fake data force the ML model to cope with noise, which neural networks are not naturally robust to without this kind of regularization technique (Goodfellow and Bengio and Courville, 2016b).

Dropout can also be thought of as a regularization method that adds noise, though to the weights rather than to the input data. In traditional dense neural net layers, retaining activity in all neurons typically leads to overfitting (Krizhevsky and Sutskever and Hinton, 2017). However, if the output of each hidden neuron has a 50% chance of being set to 0, roughly half of the neurons will be randomly removed from the neural net. The NN must then learn with a sparser network, which

incidentally is faster to train. When training a CNN with dropout, each time a mini batch of data is input to the CNN, a different set of neurons is dropped out. Thus, the CNN has to adjust model weights without relying on every neuron each time. This effectively turns one CNN into an ensemble of CNNs, which generalizes better to new data (Goodfellow and Bengio and Courville, 2016b). At inference time (when applying the trained model to new data), all neurons are used; none are dropped out. Dropout is a form of bootstrap aggregation (or bagging—a method of training many NNs to create a model ensemble that is more generalizable) that is less computationally expensive, and more effective at reducing generalization error, than an ensemble of separately trained NNs. The tradeoff is that dropout requires a large model to implement and, if the dataset is very large, dropout doesn't reduce generalization error very much. While dropout is less expensive than a literal ensemble, it still comes with the added cost of a larger model. If the dataset is large, dropout may not be worthwhile (Goodfellow and Bengio and Courville, 2016b).

S1.4 Hyperparameter Experiments

CNNs have many hyperparameters: model settings that must be pre-determined by the user and cannot be adjusted during training. Hyperparameters are decision points that can have a profound influence on model performance. Some hyperparameters can be chosen based on prior experimental work, if they have been found to be less influential or to work robustly across model applications. An example of this is the slope parameter of the leaky ReLU activation function. As long as some non-linearity is introduced to the model, the precise degree by which the negative values are attenuated does not matter (Lagerquist and Mcgovern and Gagne Ii, 2019). Pre-determining some hyperparameters or reducing the domain of tested values for each experimental hyperparameter is one way in which limited computing resources can be allocated strategically.

We performed three hyperparameter experiments to strategically narrow the number of combinations of hyperparameters to test: two early experiments and a later experiment, reported in greater detail in this manuscript. The first early experiment involved data-augmentation settings. As noted in the previous section, extensive empirical and theoretical work has shown that well-designed data augmentation improves model robustness/generalization by exposing the model to plausible input variations, thereby reducing overfitting to spurious features or noise (Krizhevsky and Sutskever and Hinton, 2017; Perez and Wang, 2017). We tuned the data augmentation hyperparameters first by performing a grid search over noise level = {0.1, 0.2, 0.3, 0.4, 0.5}; and number of augmentations = {2, 3, 4, 5, 6, 7, 8, 9, 10}. We determined that the optimal hyperparameters settings are 0.2 and 8, the values reported in the manuscript. In this experiment, performance on the unaugmented validation data was more sensitive to noise level than to the number of augmentations. With a smaller noise level (0.1), performance on the validation data deteriorated, suggesting that 0.1 is not quite enough noise to prevent the model from overfitting to spurious features. With a larger noise level (0.3-0.5), performance on the validation data again deteriorated, suggesting that 0.3-0.5 is too much noise, overwhelming useful patterns in the data. Note that, whenever we do data augmentation, Gaussian noise is applied to the normalized predictors (in z-score units) rather than the unnormalized predictors (in physical units). Thus, a Gaussian noise level of 0.2 has an equivalent effect on all predictor variables.

The other early hyperparameter experiment involved regularization settings, in which we experimented with the optimal values for L_2 weight, dropout rate, and the number of dense layers. Specifically, we performed a grid search over L_2 weight = $\{10^{-7}, 10^{-6.5}, 10^{-6}, 10^{-5.5}, 10^{-5}\}$; dropout rate = {0.500, 0.575, 0.650, 0.725, 0.800}; and number of dense layers = {3, 4, 5, 6, 7}. We trained a separate model for all 5 x 5 x 5 = 125 combinations of these three hyperparameters. We made the following conclusions from this early experiment:

- The optimal number of dense layers was 3 or 4, while values of 5-7 led to overfitting. Since 3 and 4 are on the edge of the search space, for the experiment in the manuscript, we decided to experiment more widely, also training models with 1-2 dense layers.
- The optimal dropout rate was 0.575 or 0.650.
- The optimal L_2 weight was $10^{-6.5}$ or 10^{-6}.

While the effective sample size remains limited, our data-augmentation strategy contributes meaningful diversity to the training data that complements regularization techniques. The combination of data augmentation and regularization enhances model robustness/performance in a way not achievable by regularization alone. Based on these results, we proceeded with the third hyperparameter as described in section 2.5.3 of the manuscript.

S2 Hyperparameter Experiment Results

Each subsection below shows the results from one hyperparameter experiment, used to determine the best six-class, four-class, or two-class model.

S2.1 Six-Class Model Hyperparameter Results

Figure 5 of the main body shows top-1 accuracy for every hyperparameter combination in the six-class experiment. Figures S1-S7 are analogous, showing results for the other seven evaluation metrics.

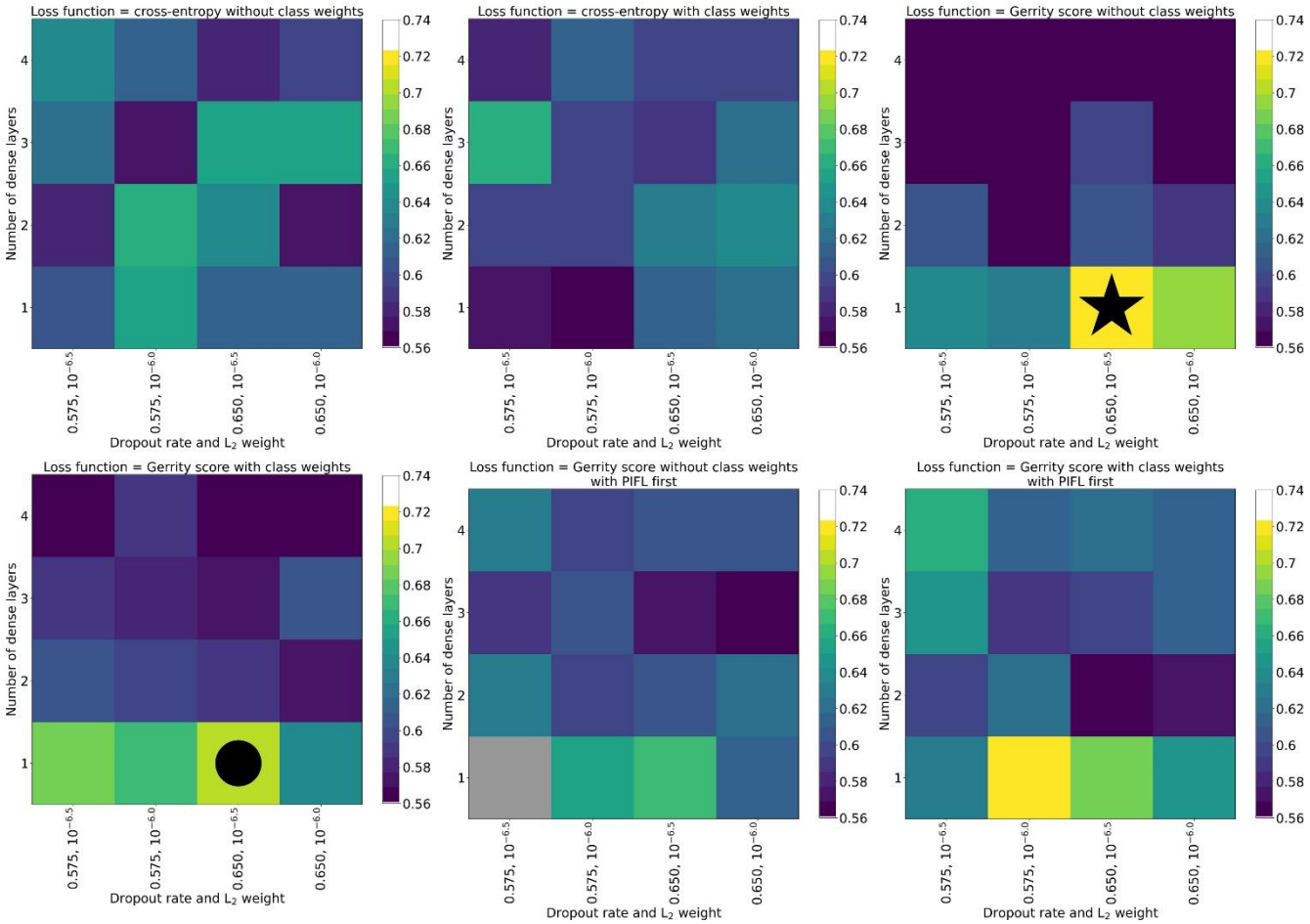


Figure S1: Hyperparameter experiment results for the six-class classification, evaluated by top-2 accuracy. Formatting is described in the caption of Figure 5 in the main body.

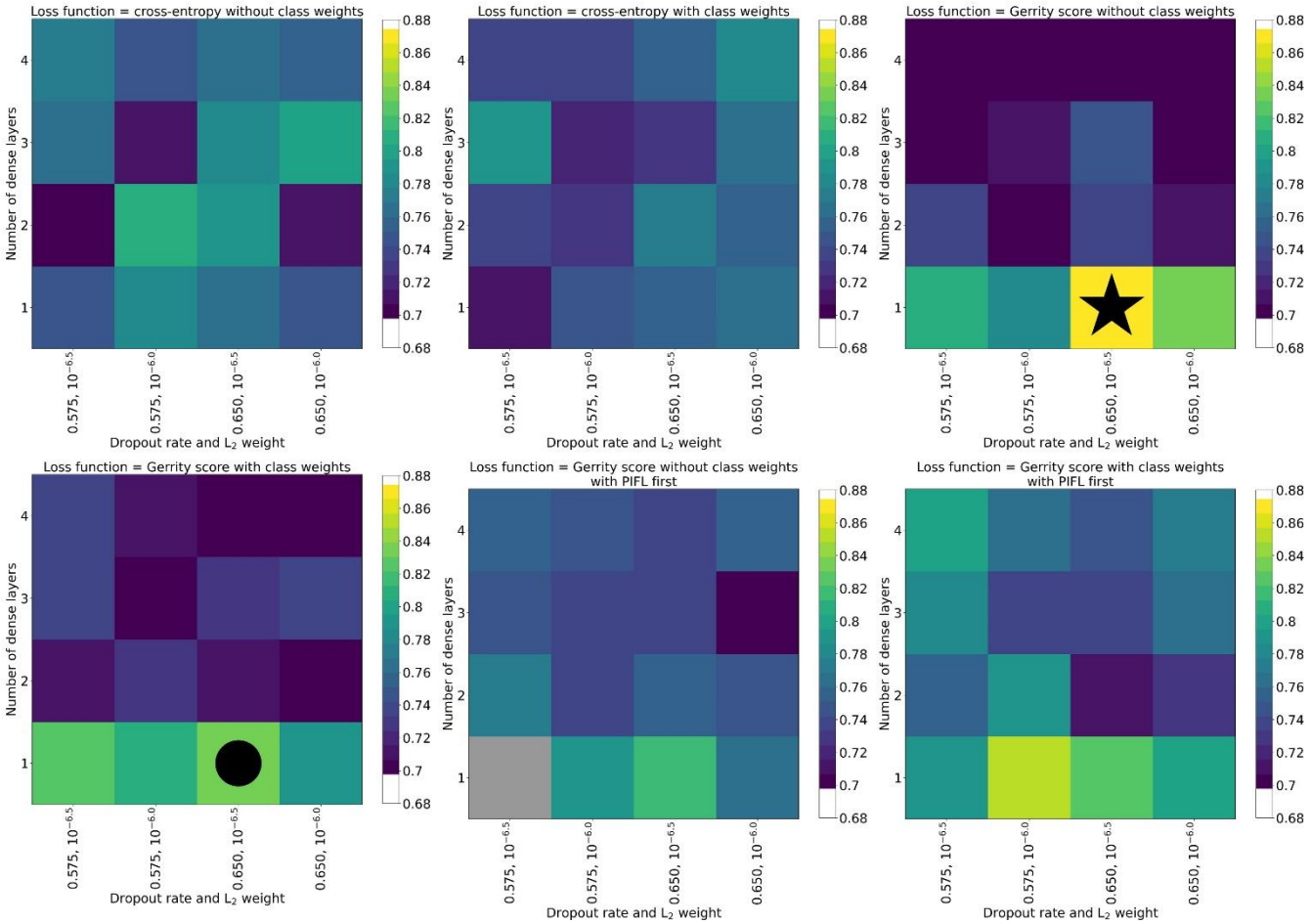


Figure S2: Hyperparameter experiment results for the six-class classification, evaluated by top-3 accuracy. Formatting is described in the caption of Figure 5 in the main body.

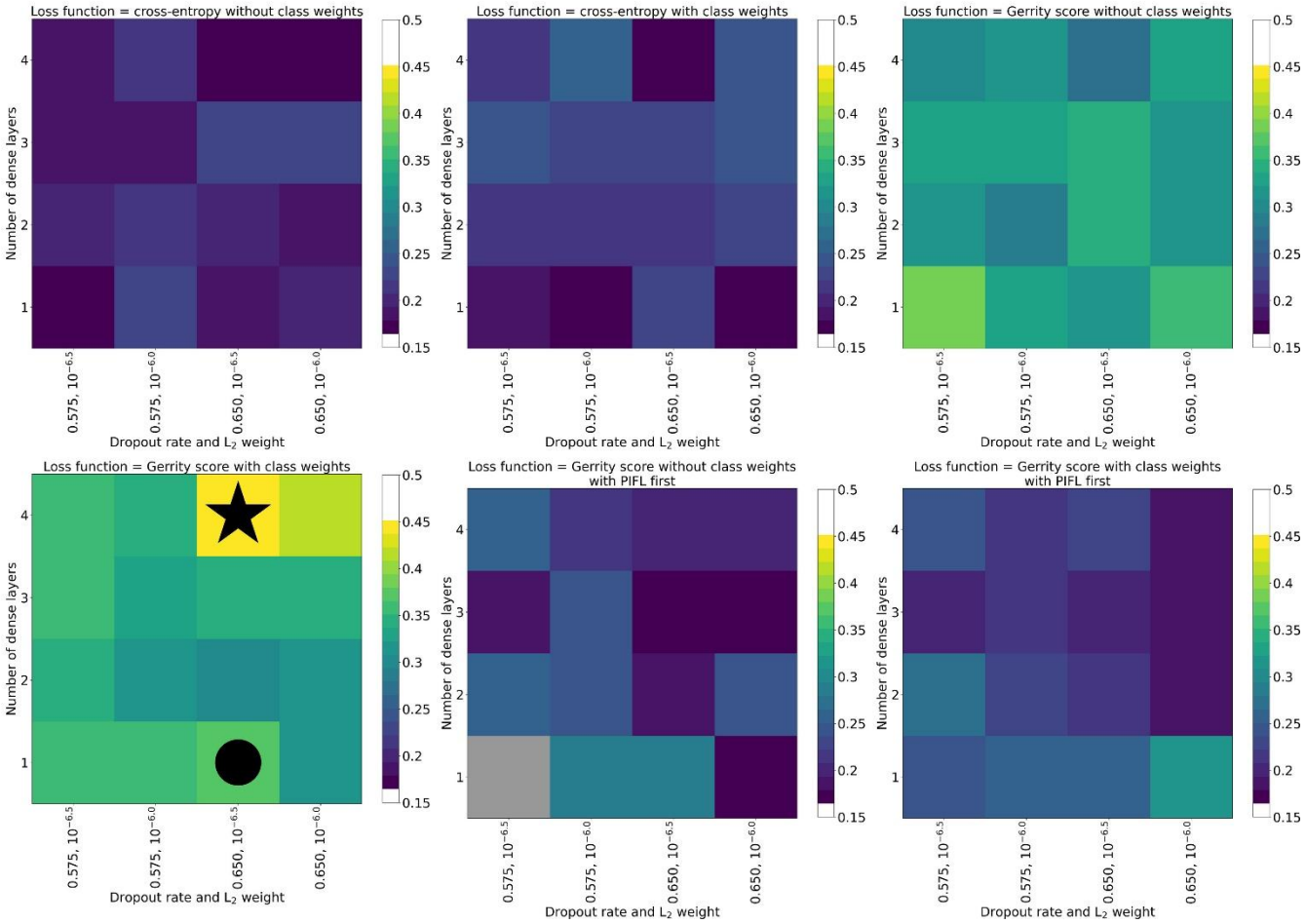
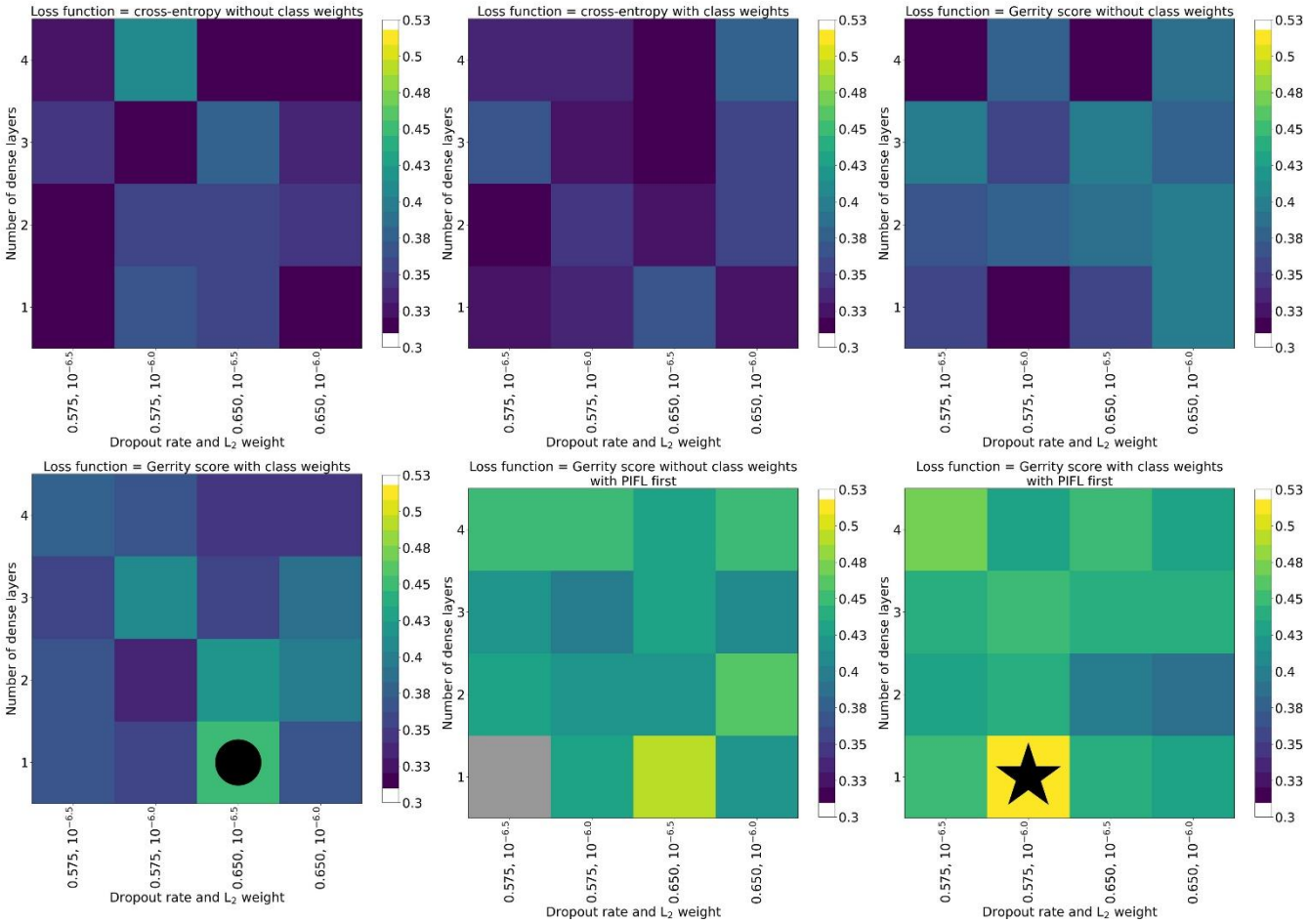
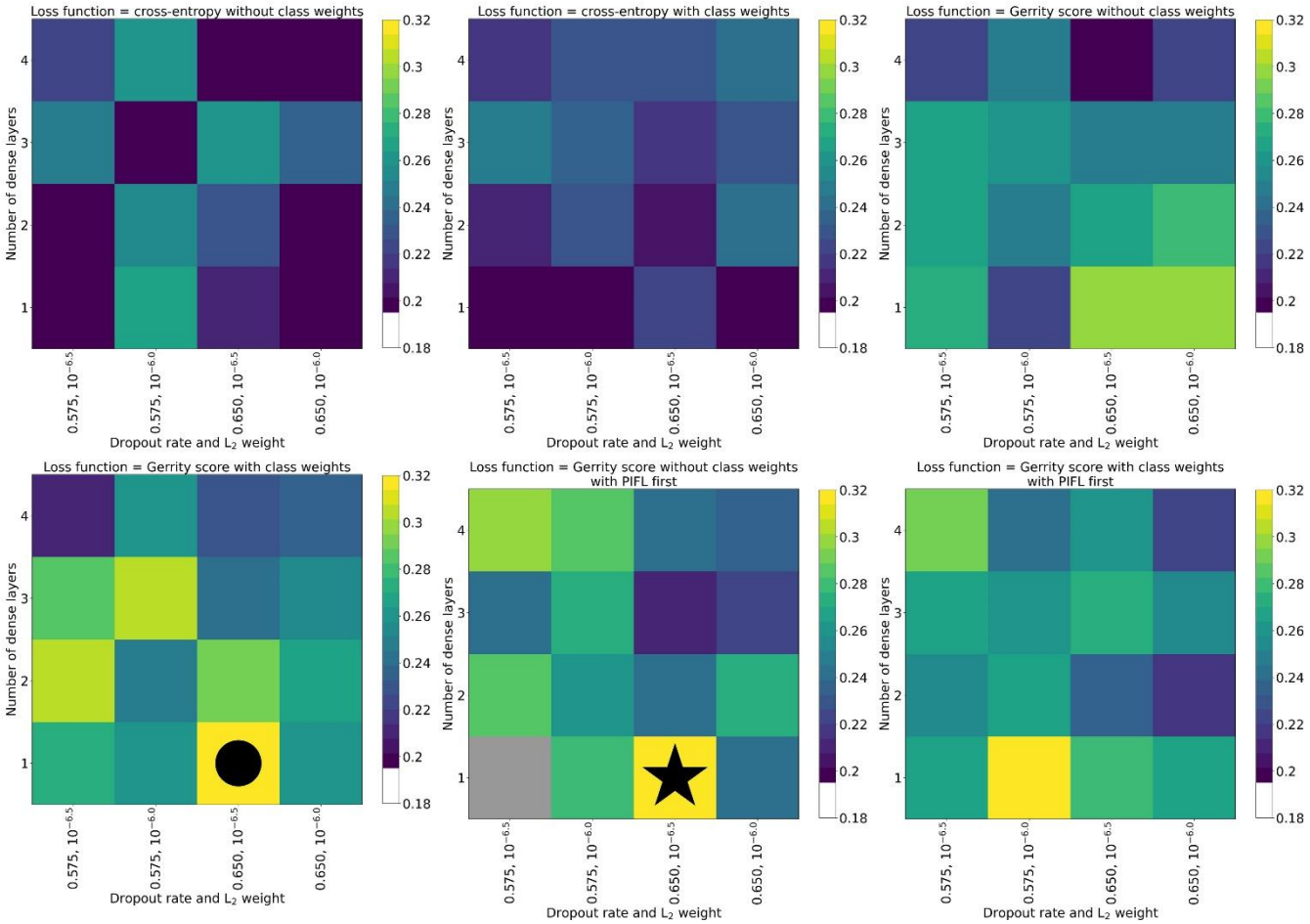


Figure S3: Hyperparameter experiment results for the six-class classification, evaluated by the Gerrity score. Formatting is described in the caption of Figure 5 in the main body.



185

Figure S4: Hyperparameter experiment results for the six-class classification, evaluated by the PIFL-first Gerrity score. Formatting is described in the caption of Figure 5 in the main body.



190 **Figure S5: Hyperparameter experiment results for the six-class classification, evaluated by the Heidke score. Formatting is described in the caption of Figure 5 in the main body.**

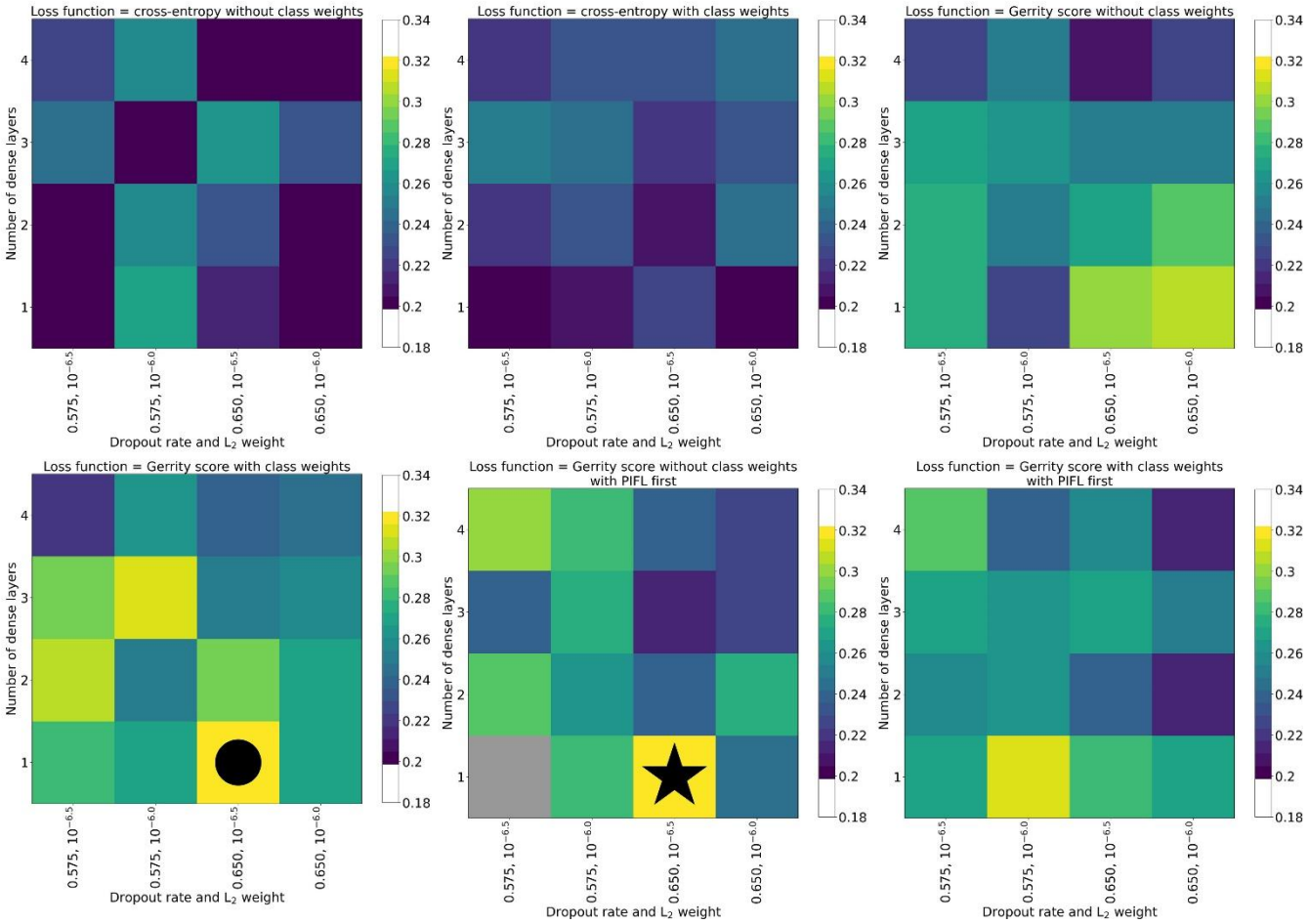


Figure S6: Hyperparameter experiment results for the six-class classification, evaluated by the Peirce score. Formatting is described in the caption of Figure 5 in the main body.

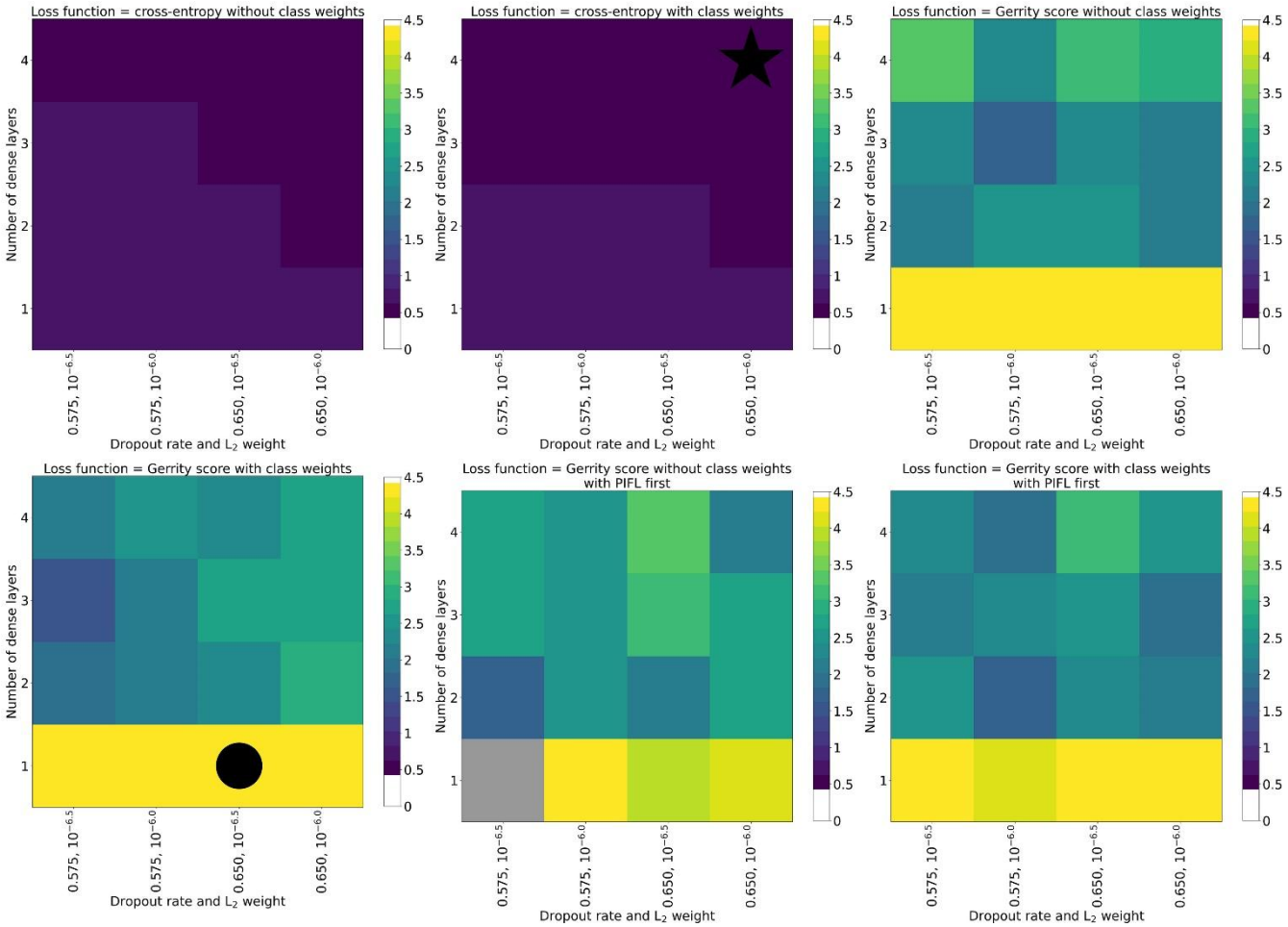
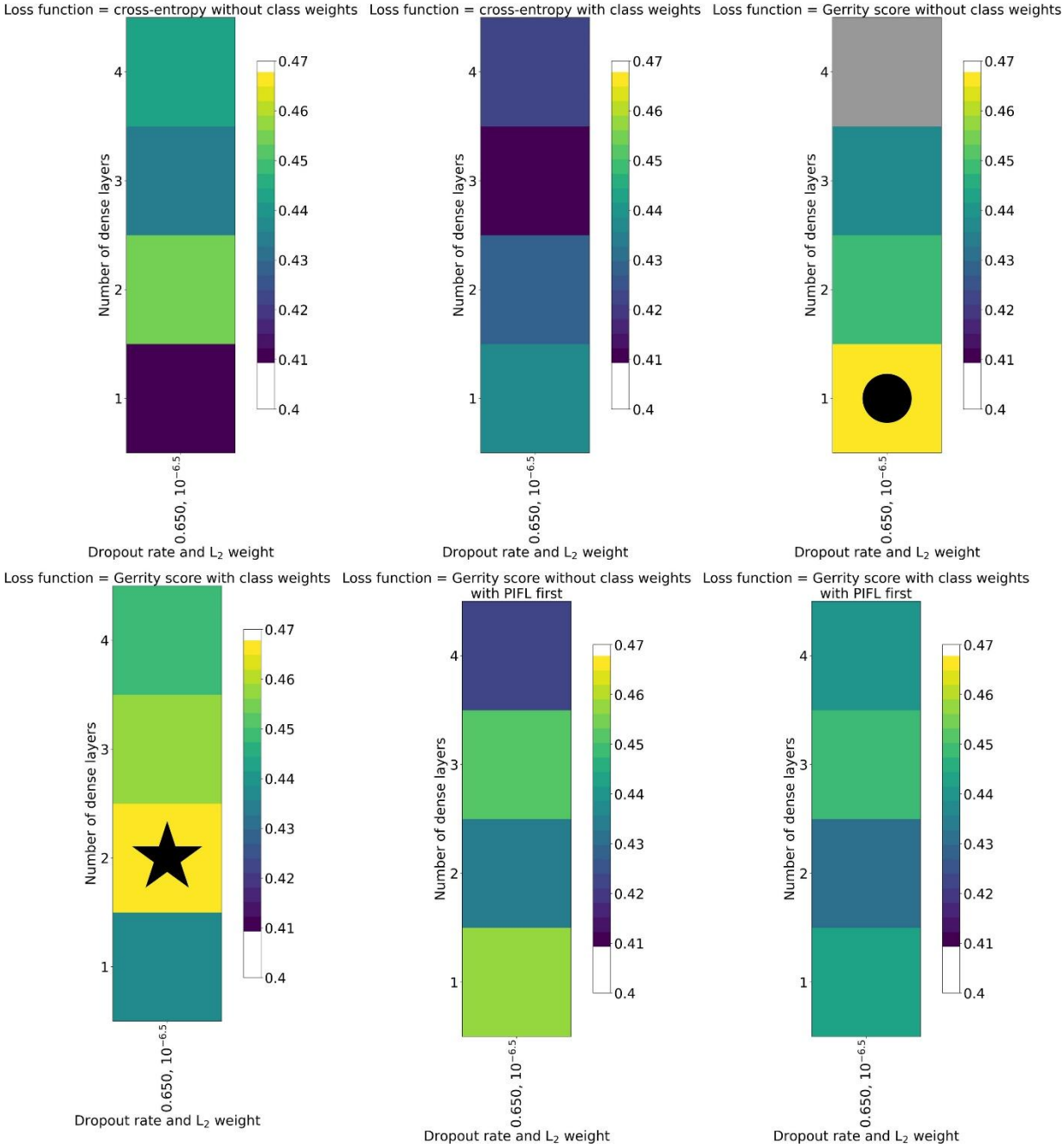


Figure S7: Hyperparameter experiment results for the six-class classification, evaluated by cross-entropy. Formatting is described in the caption of Figure 5 in the main body.

S2.2 Four-Class Model Hyperparameter Results

Figures S8-S15 show results for the four-class hyperparameter experiment.



205 **Figure S8: Hyperparameter experiment results for the four-class classification, evaluated by top-1 accuracy. Formatting is described in the caption of Figure 5 in the main body.**

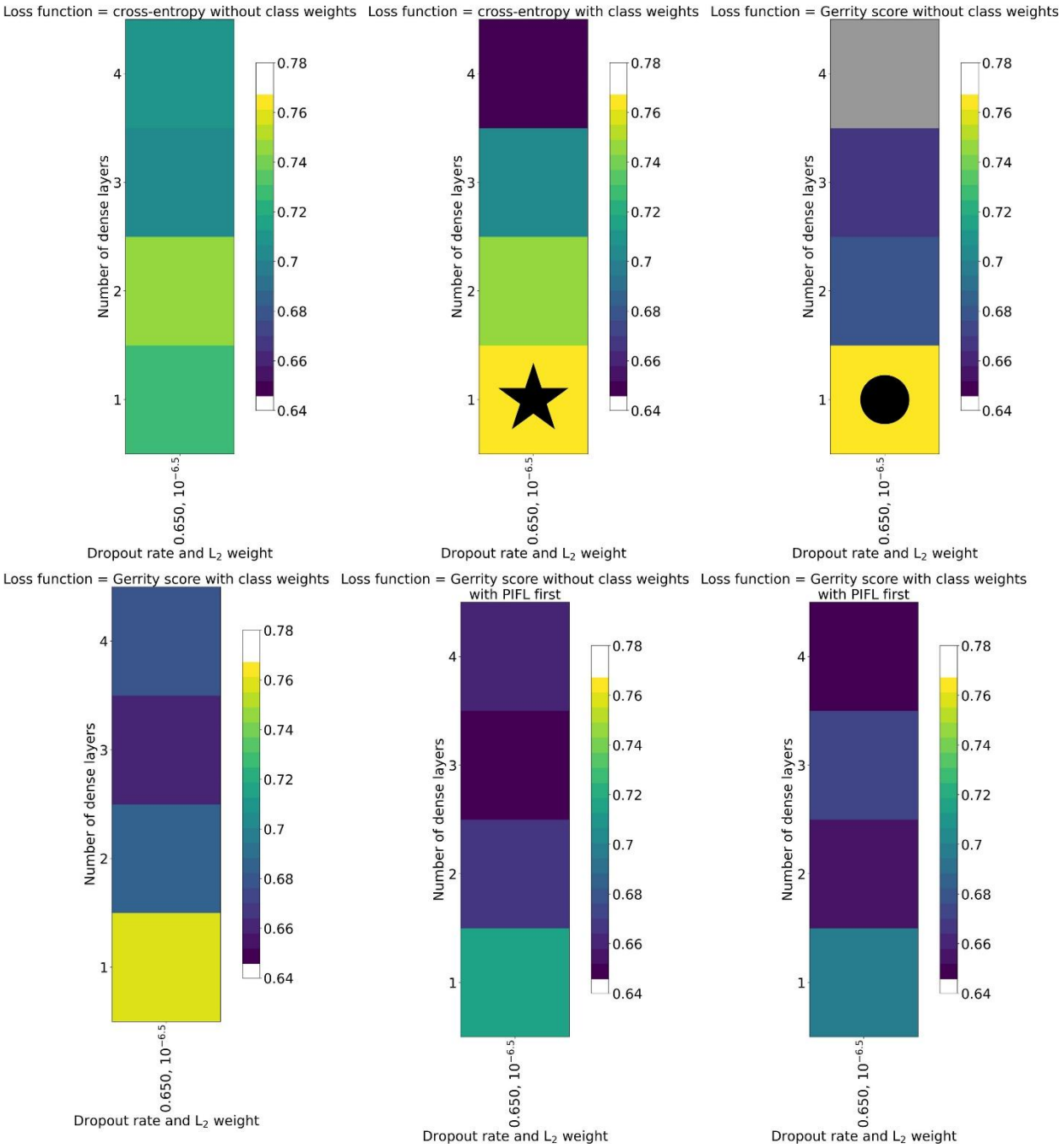


Figure S9: Hyperparameter experiment results for the four-class classification, evaluated by top-2 accuracy. Formatting is described in the caption of Figure 5 in the main body.

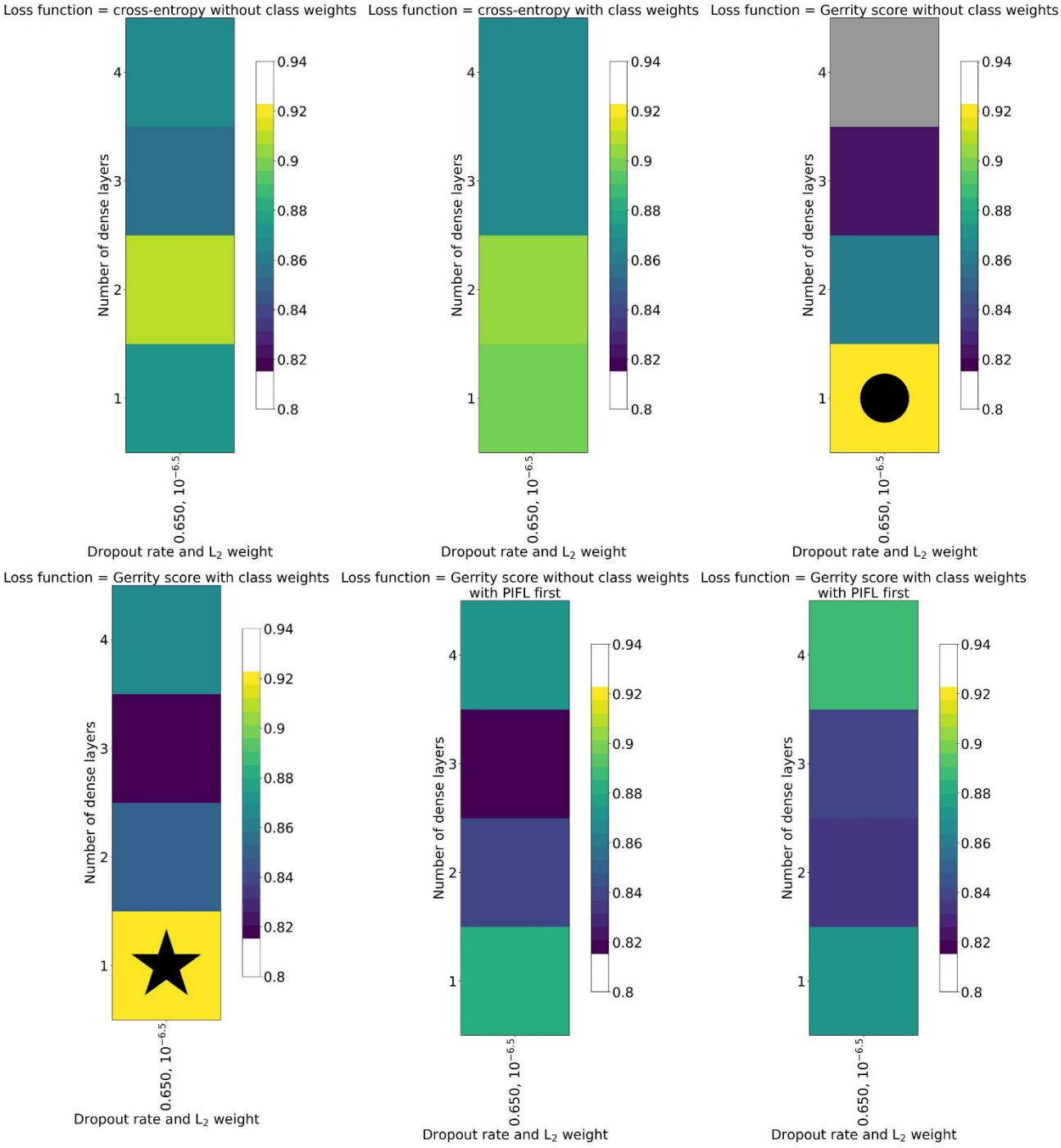
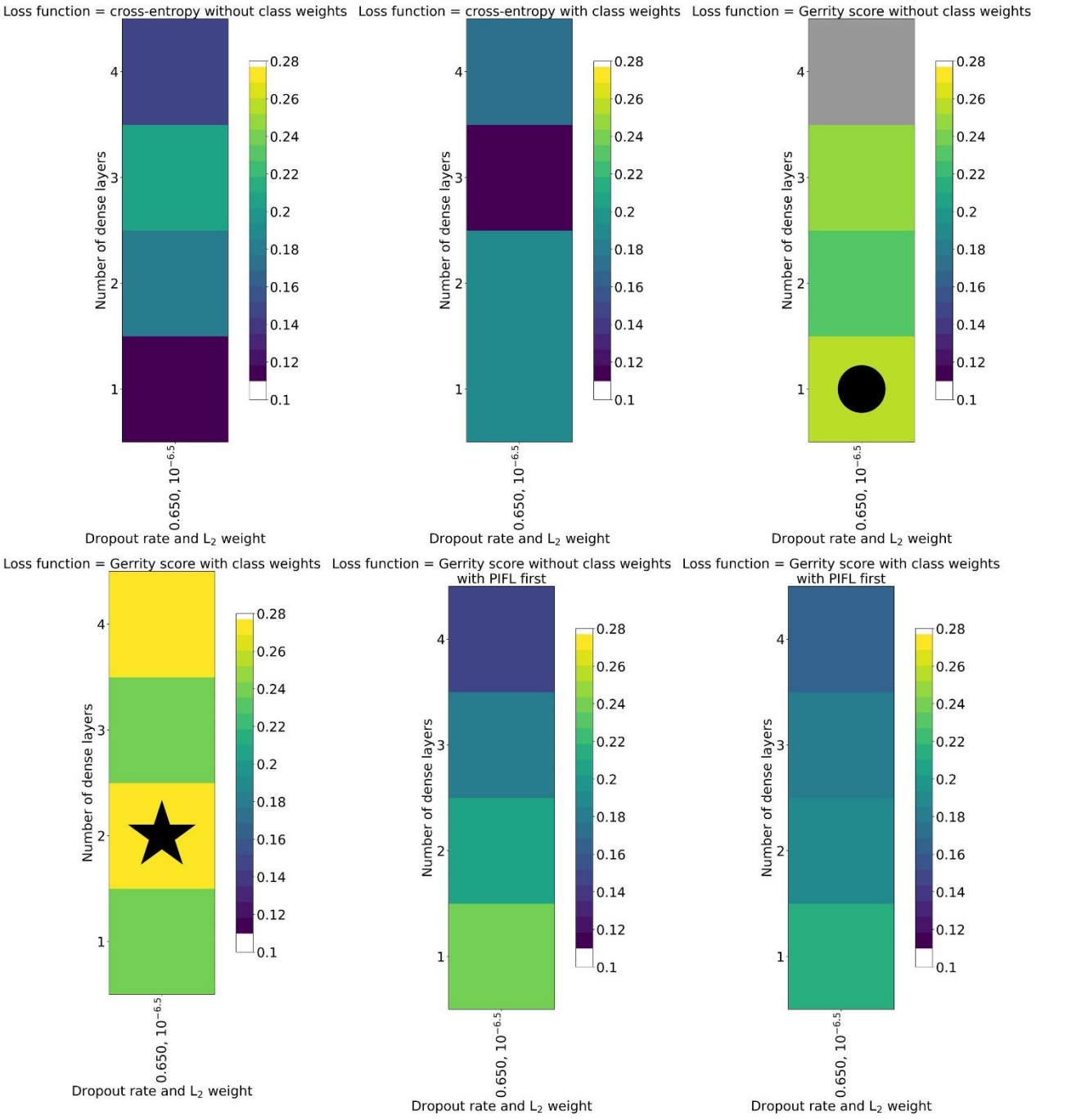


Figure S10: Hyperparameter experiment results for the four-class classification, evaluated by top-3 accuracy. Formatting is described in the caption of Figure 5 in the main body.



215 **Figure S11: Hyperparameter experiment results for the four-class classification, evaluated by the Gerrity score. Formatting is described in the caption of Figure 5 in the main body.**

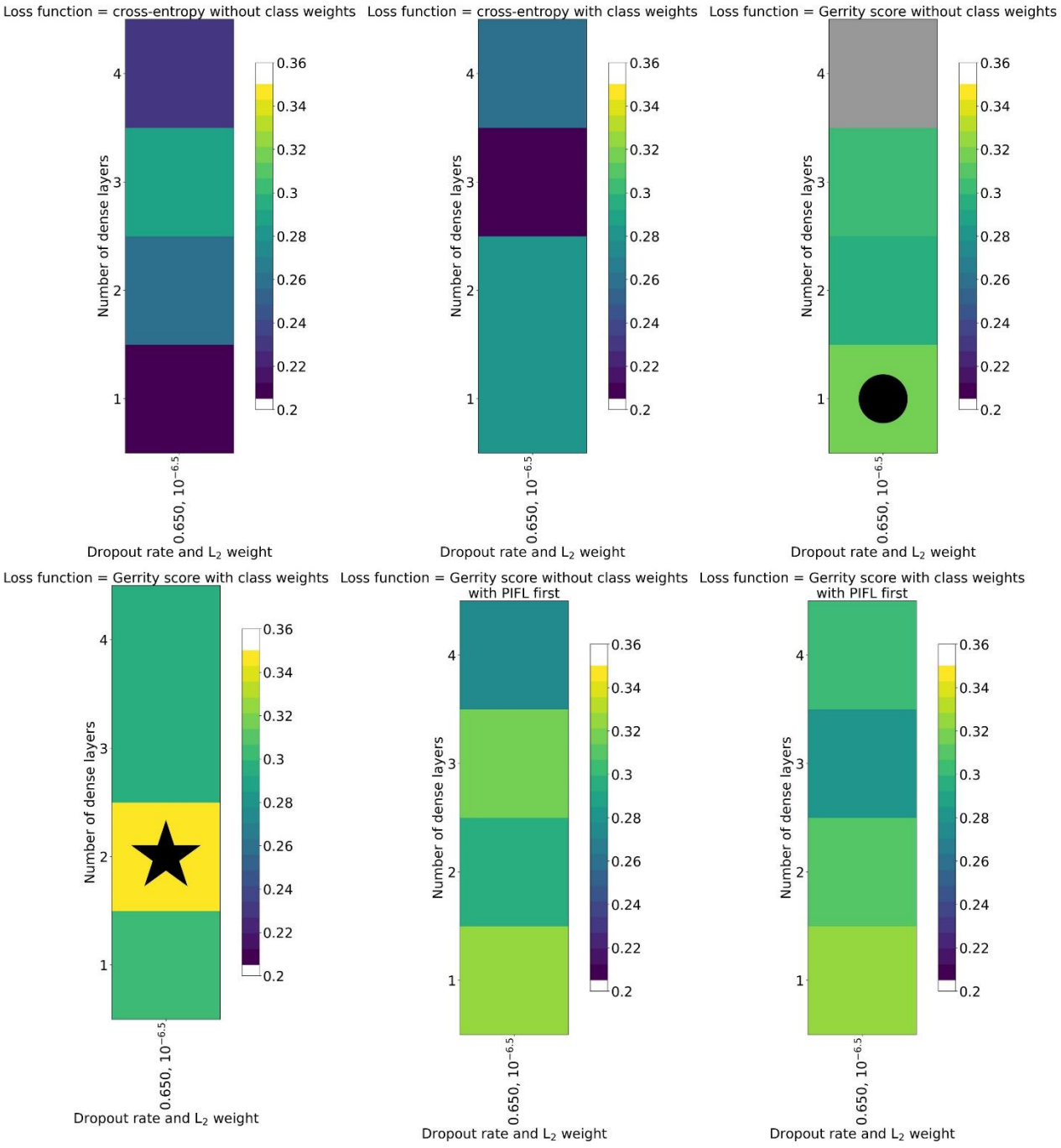
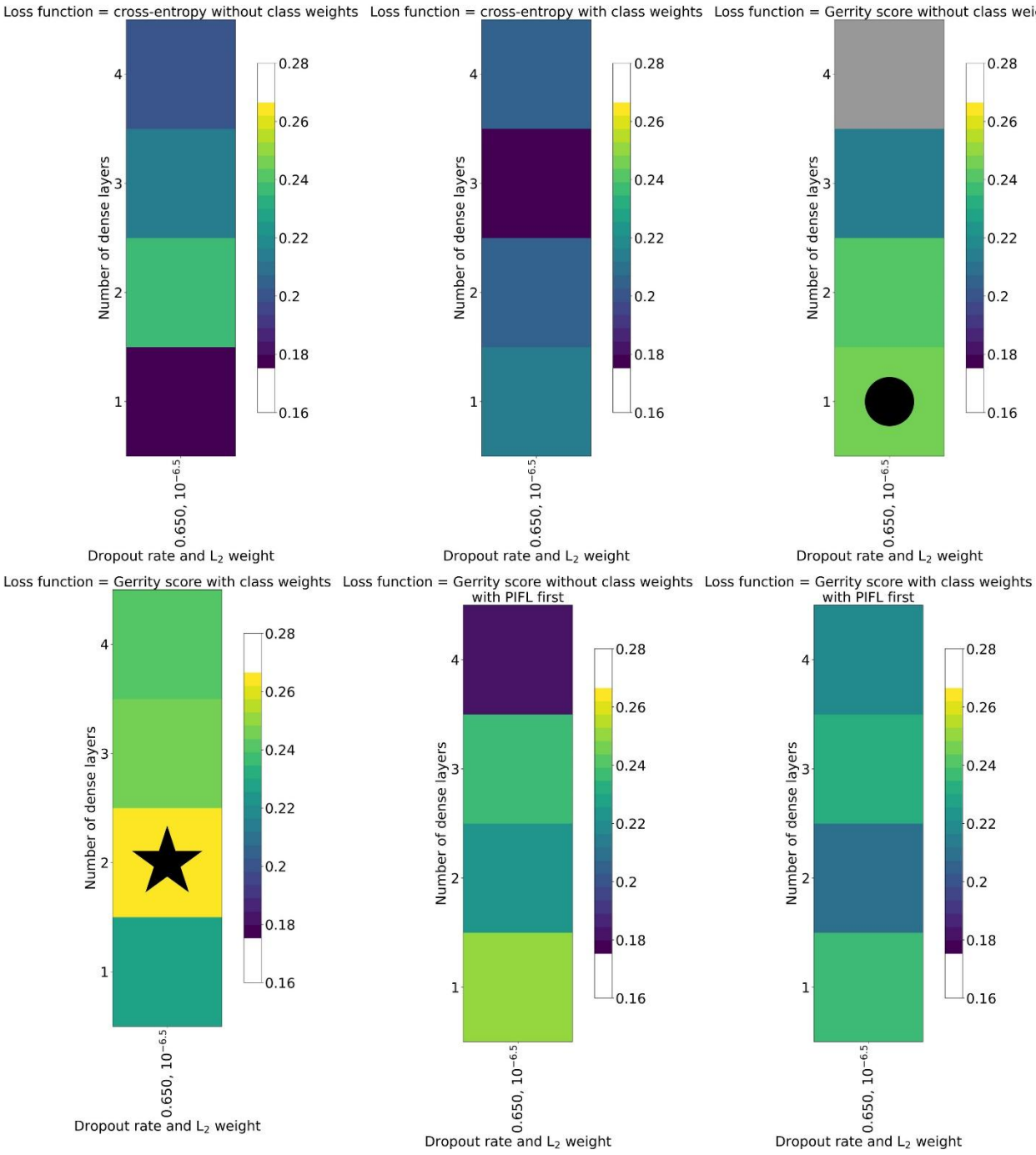


Figure S12: Hyperparameter experiment results for the four-class classification, evaluated by the PIFL-first Gerrity score. Formatting is described in the caption of Figure 5 in the main body.



220

Figure S13: Hyperparameter experiment results for the four-class classification, evaluated by the Heidke score. Formatting is described in the caption of Figure 5 in the main body.

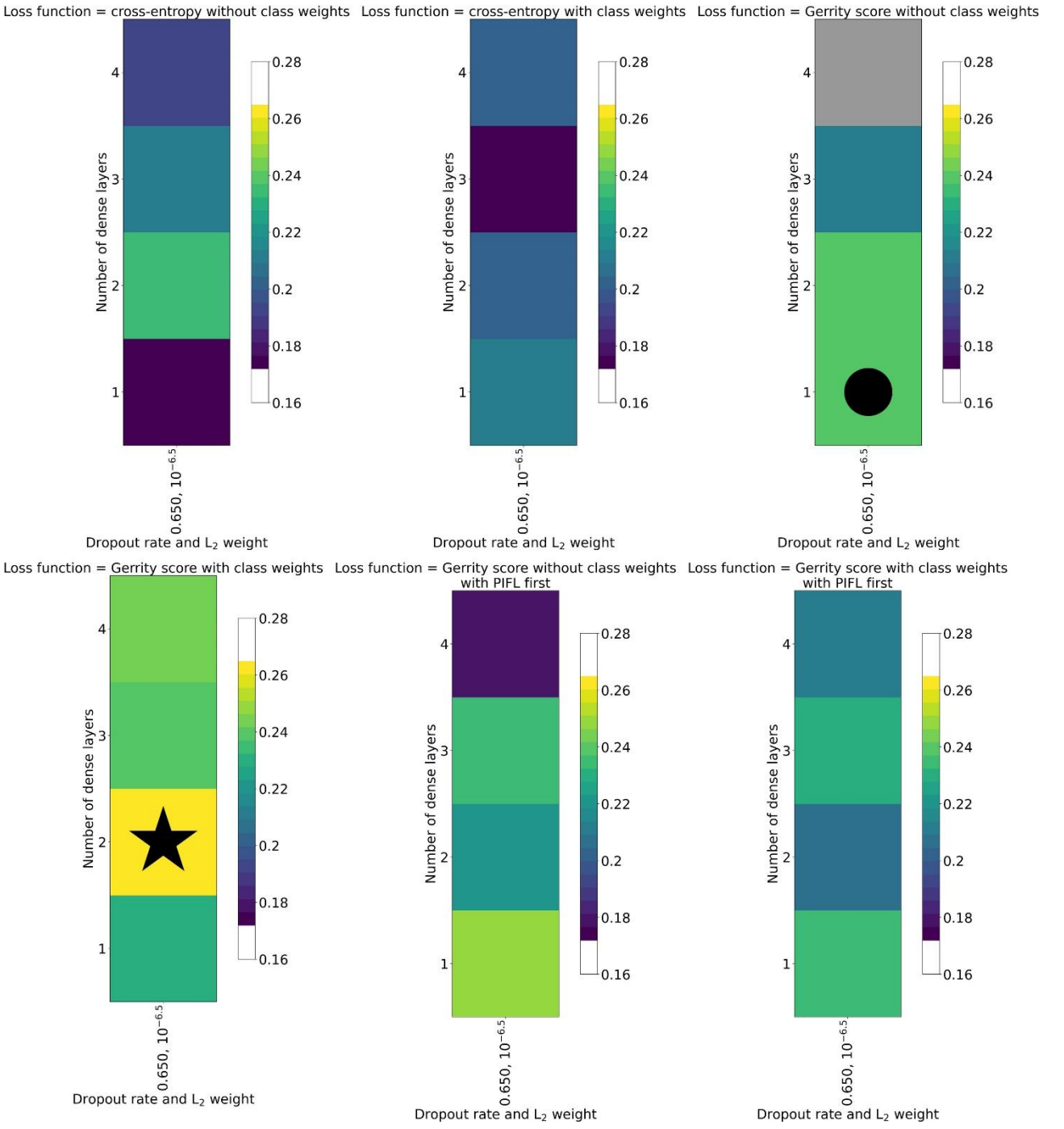


Figure S14: Hyperparameter experiment results for the four-class classification, evaluated by the Peirce score. Formatting is described in the caption of Figure 5 in the main body.

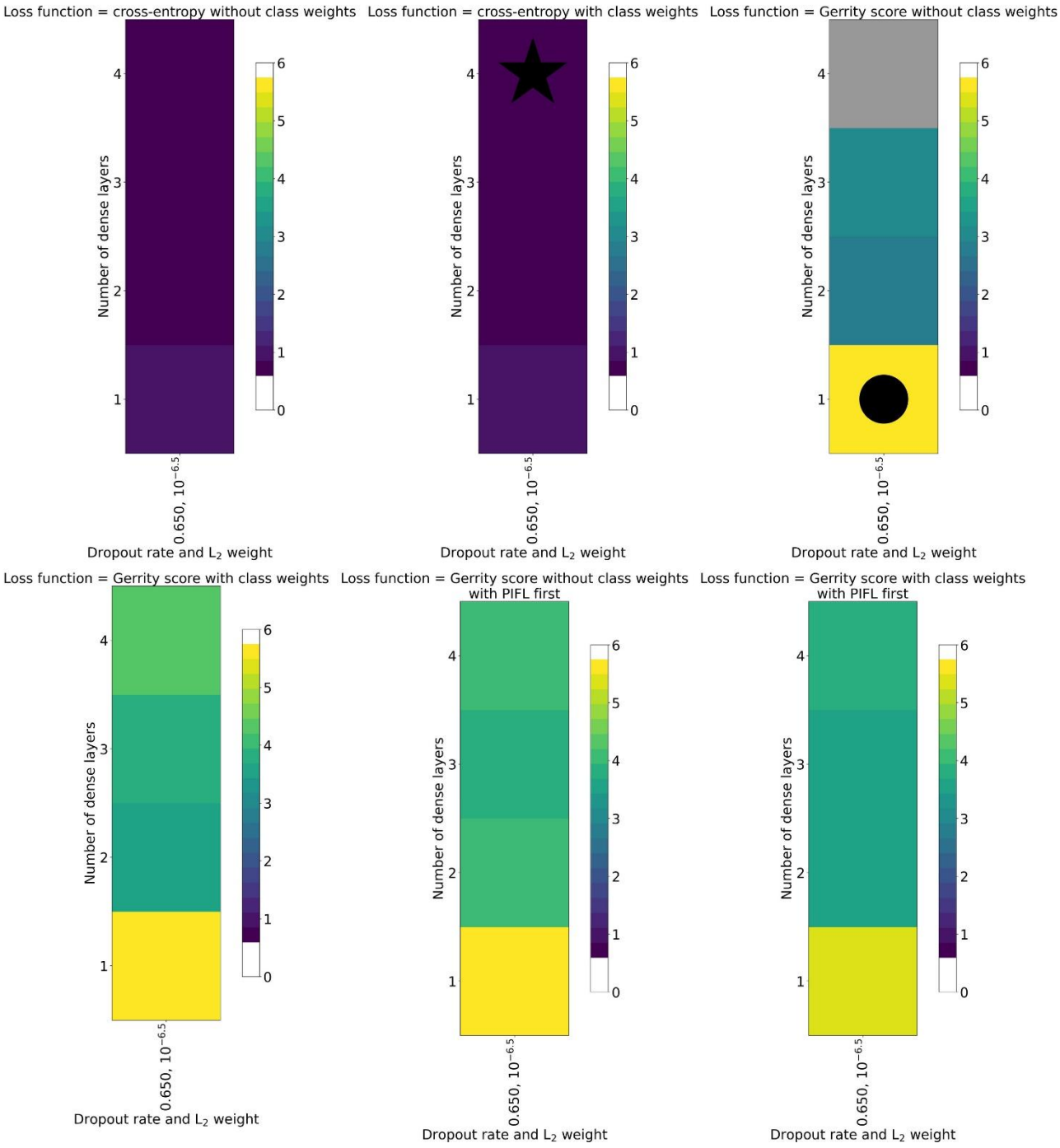


Figure S15: Hyperparameter experiment results for the four-class classification, evaluated by cross-entropy. Formatting is described in the caption of Figure 5 in the main body.

S2.3 Two-Class Model Hyperparameter Results

Figures S16-S21 show results for the two-class hyperparameter experiment.

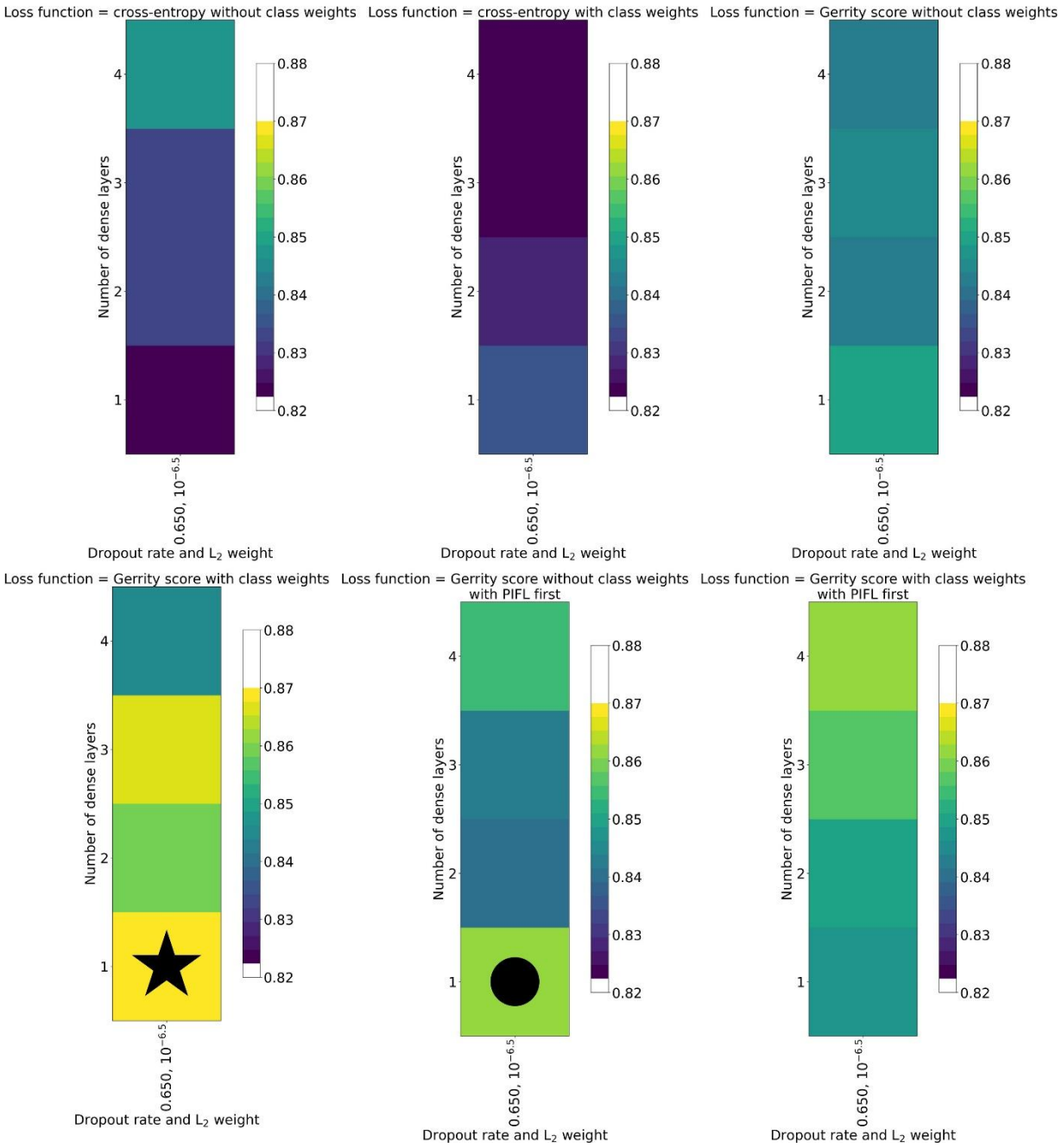


Figure S16: Hyperparameter experiment results for the two-class classification, evaluated by top-1 accuracy. Formatting is described in the caption of Figure 5 in the main body.

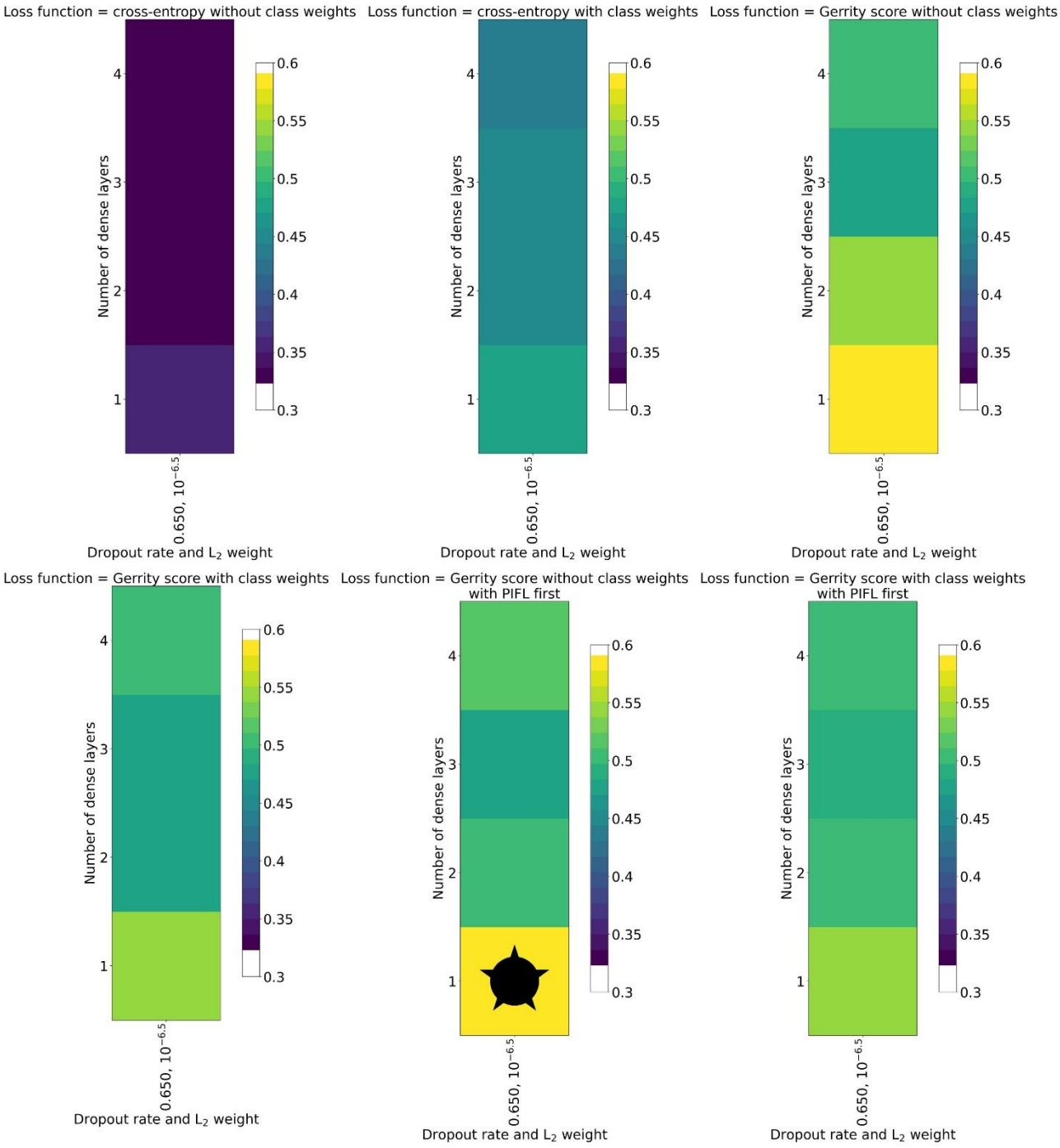


Figure S17: Hyperparameter experiment results for the two-class classification, evaluated by the Gerrity score. Formatting is described in the caption of Figure 5 in the main body.

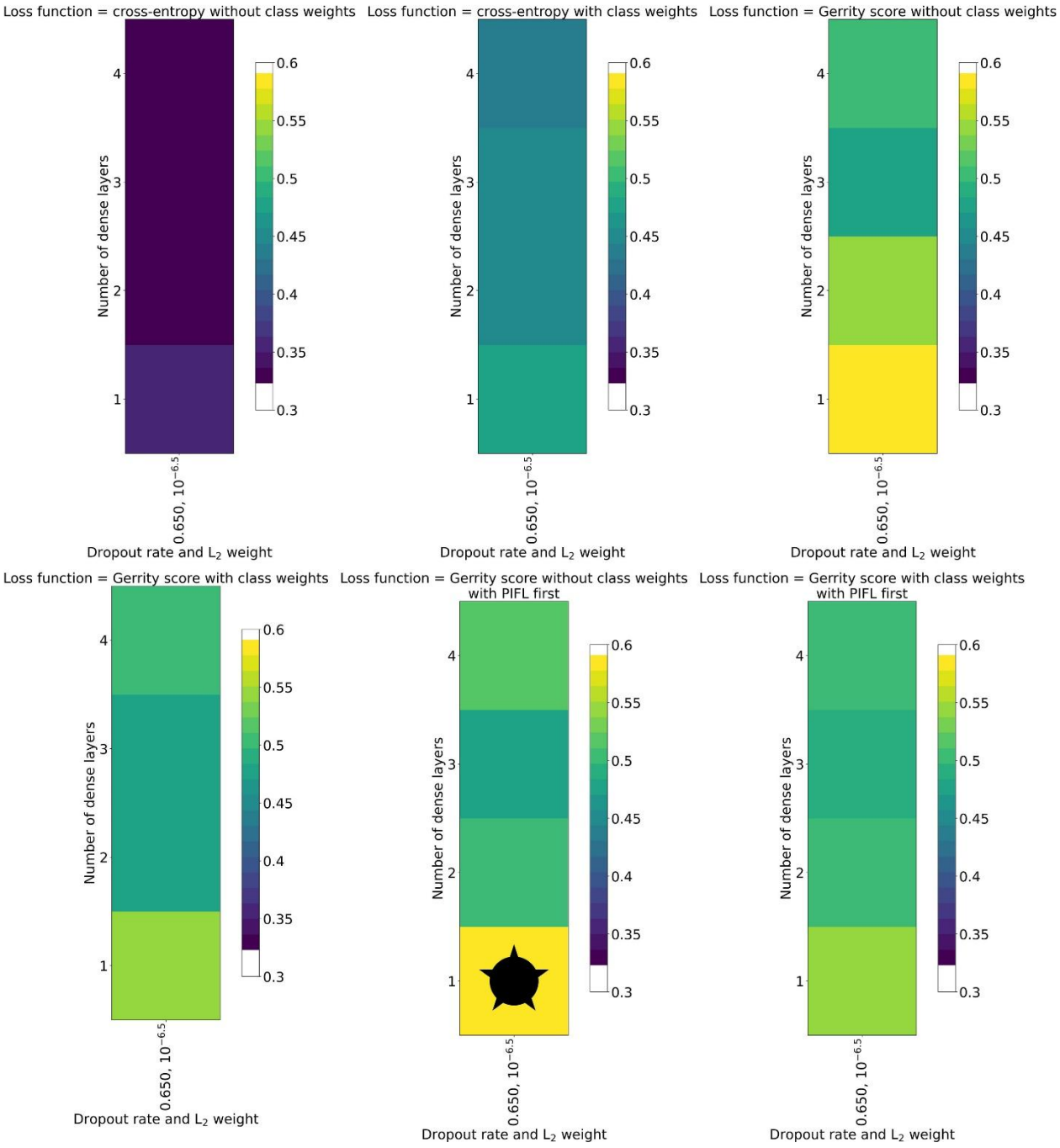


Figure S18: Hyperparameter experiment results for the two-class classification, evaluated by the PIFL-first Gerrity score. Formatting is described in the caption of Figure 5 in the main body.

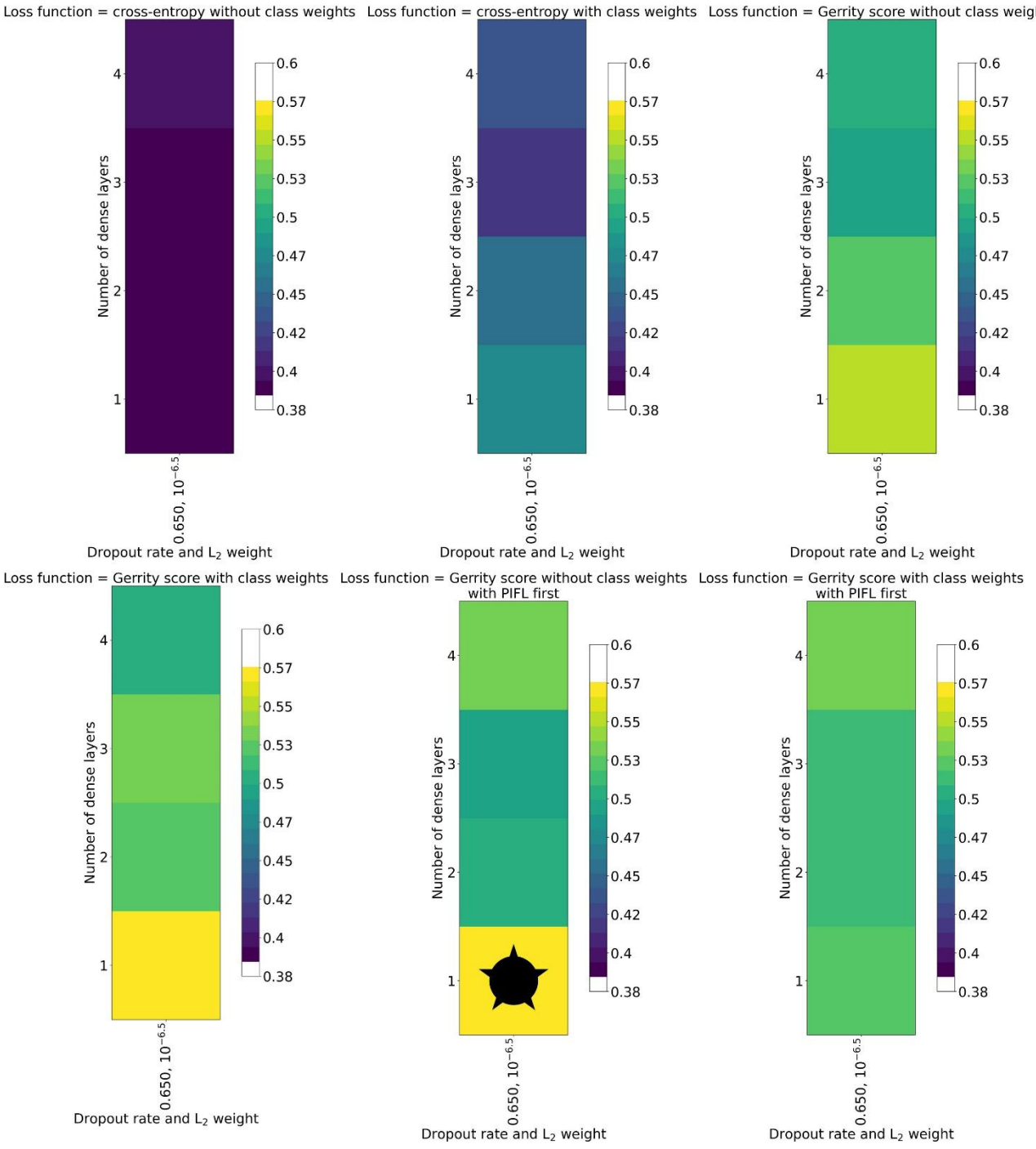
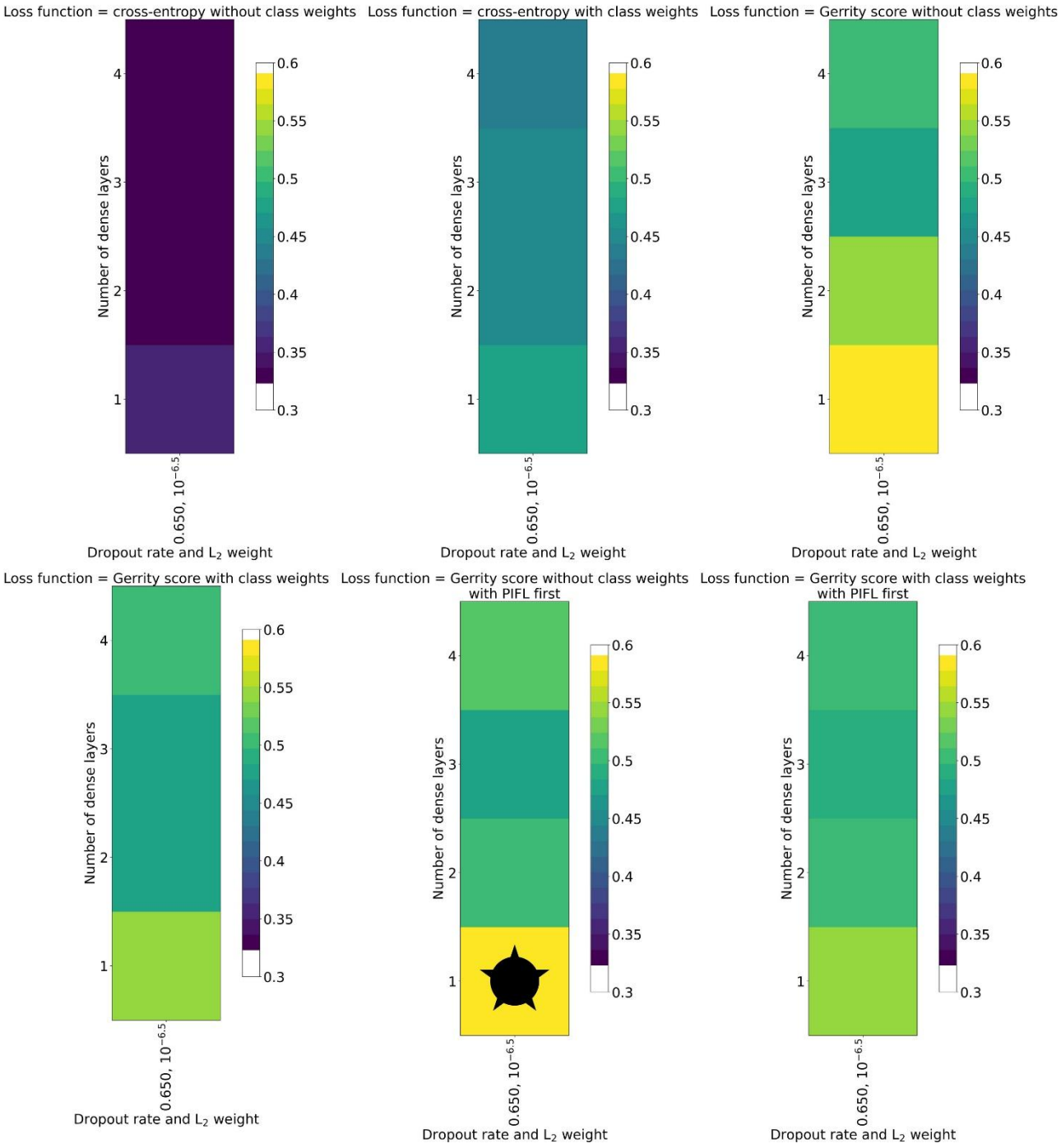


Figure S19: Hyperparameter experiment results for the two-class classification, evaluated by the Heidke score. Formatting is described in the caption of Figure 5 in the main body.



245 **Figure S20: Hyperparameter experiment results for the two-class classification, evaluated by the Peirce score. Formatting is described in the caption of Figure 5 in the main body.**

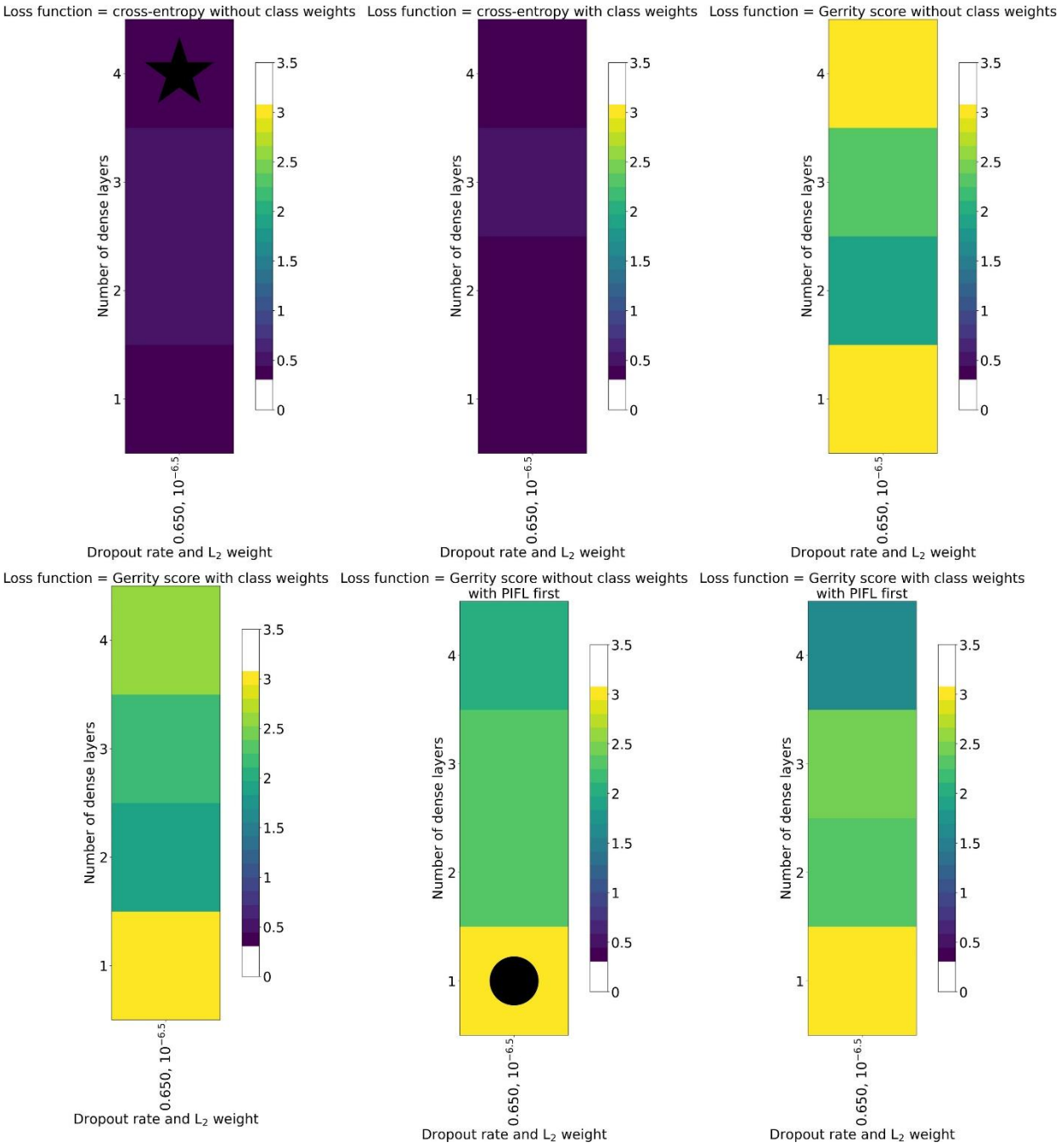


Figure S21: Hyperparameter experiment results for the two-class classification, evaluated by cross-entropy. Formatting is described in the caption of Figure 5 in the main body.

S3 Training Diagrams for Selected Models

Over training epochs, model performance improved for the selected loss function (the class-weighted Gerrity score) and improved in terms of top-1, top-2, and top-3 accuracy. While the models consistently performed better with the training data (solid lines in the below figures) than the validation data (dashed lines), the models do converge for the validation data. Without regularization (data augmentation, L2 regularization, and dropout), the validation curves do not converge in this way. However, these models may still suffer somewhat from overfitting and would be improved with additional data.

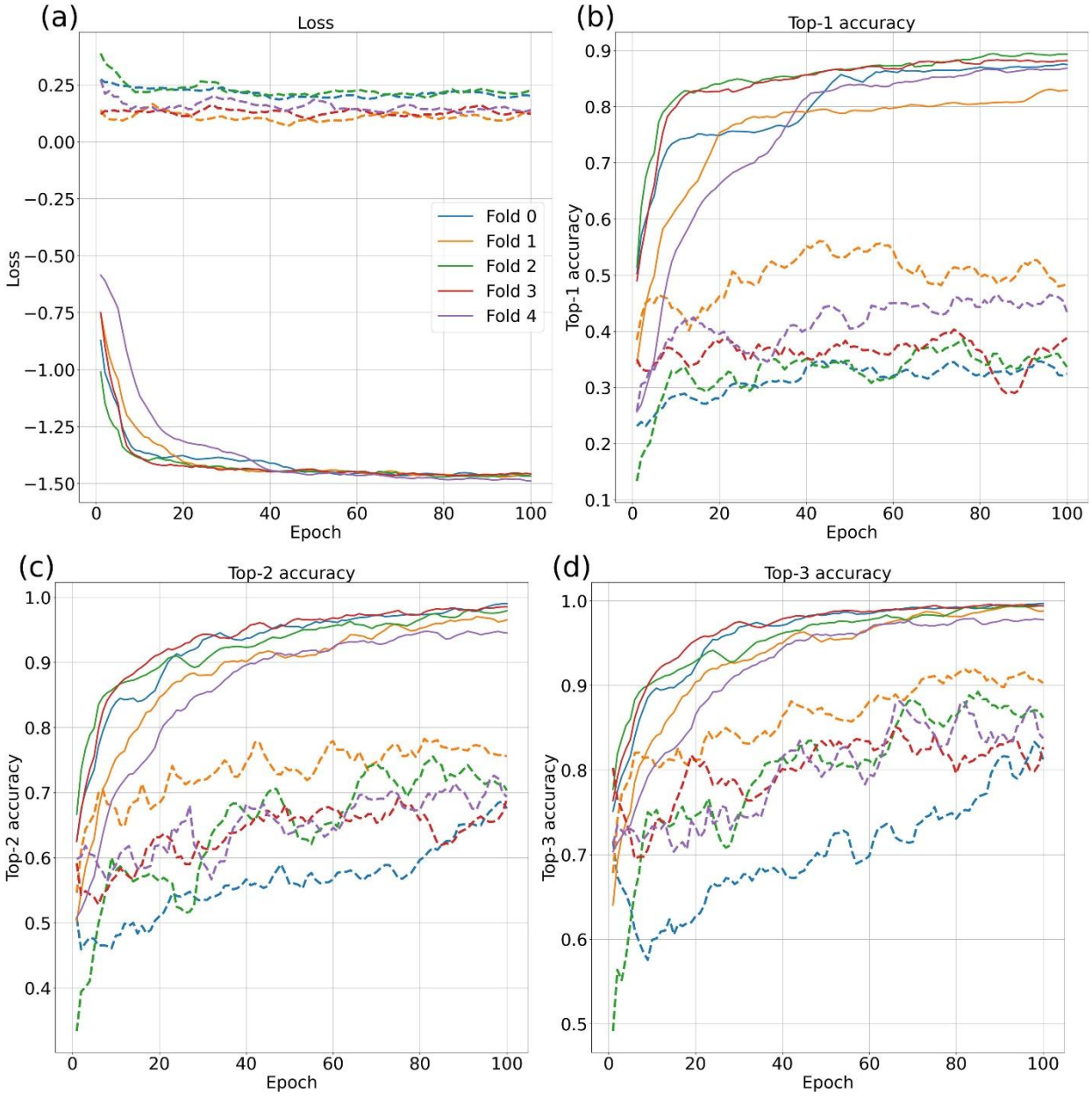


Figure S22. Learning curves for the six-class CNN model. Each panel shows the evolution of one performance metric over the 100 training epochs. Dashed (solid) lines represent the validation (training) data. Note that, because Keras expects the loss function to be negatively oriented (such that lower is better), the loss function here is actually the *negative* class-weighted Gerrity score.

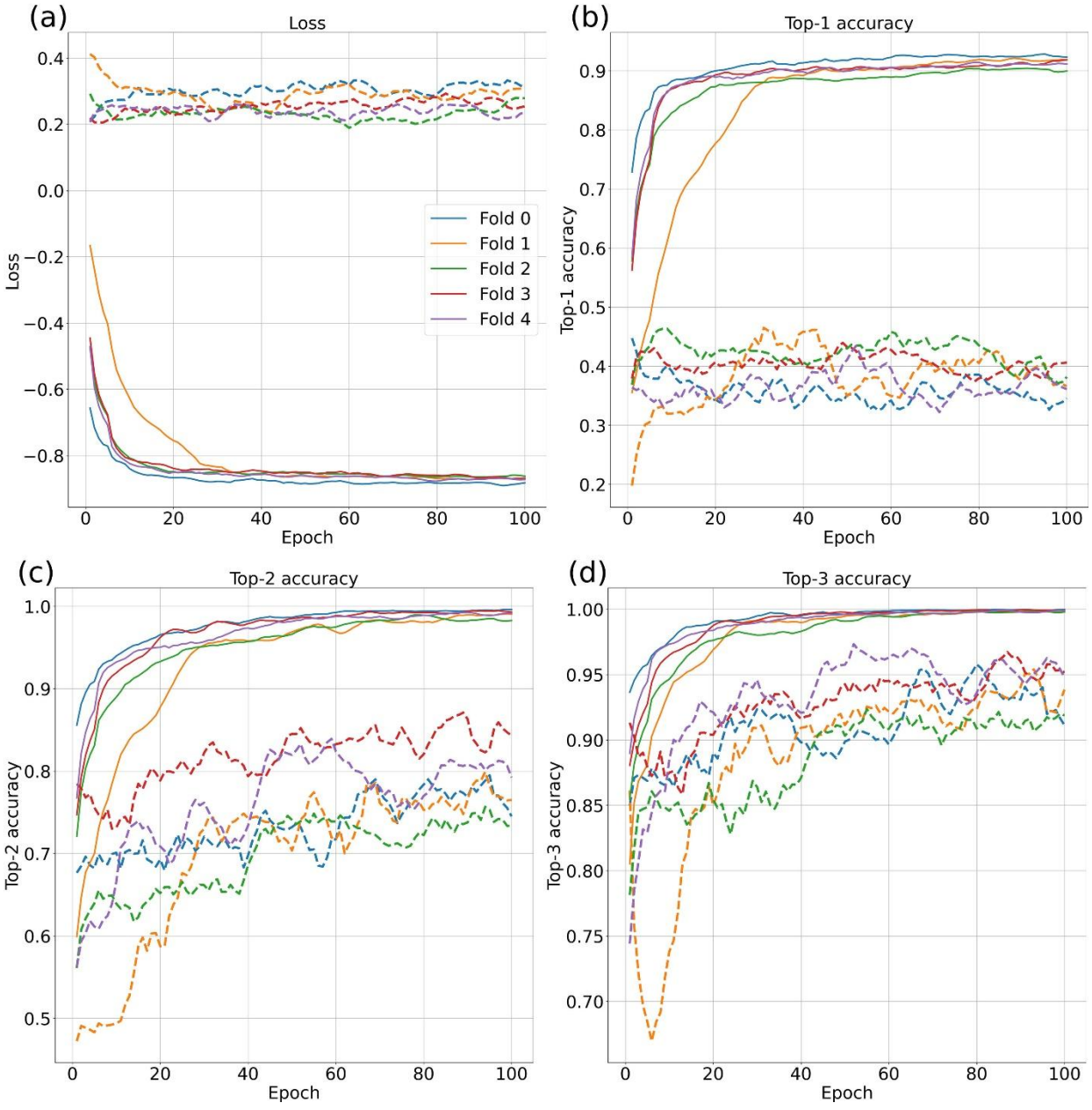


Figure S23. Learning curves for the four-class CNN model. Formatting is explained in the caption of Figure S22. Again, note that because Keras expects the loss function to be negatively oriented, the loss function here is actually the *negative* default Gerrity score.

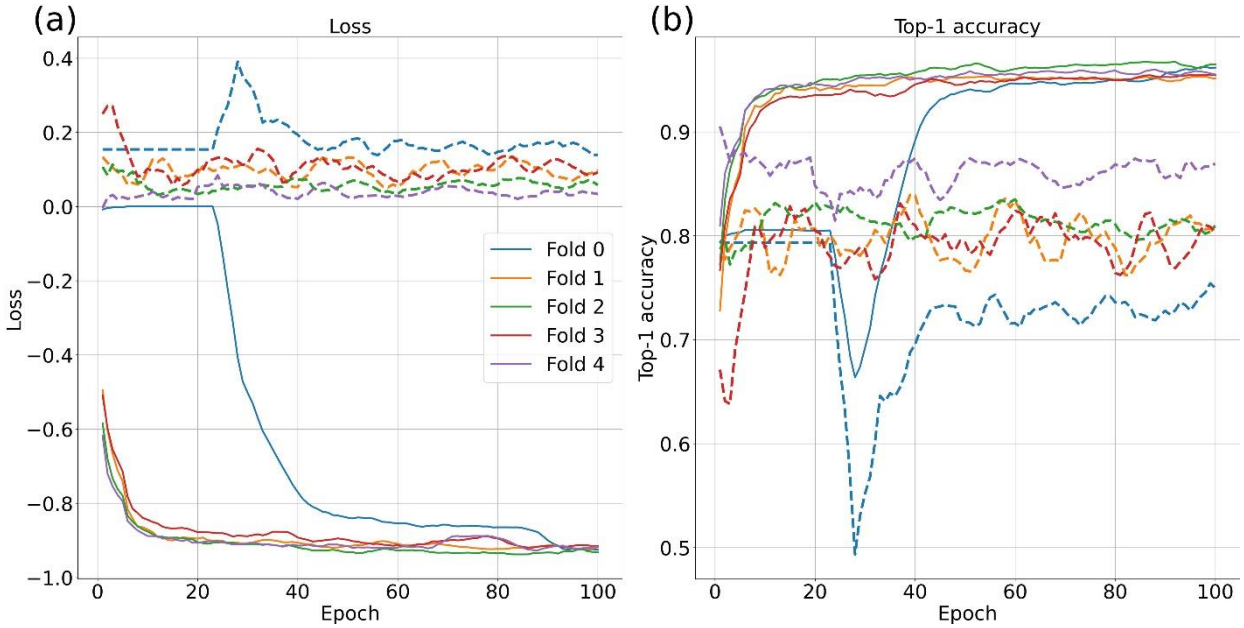


Figure S24. Learning curves for the two-class CNN model. Formatting is explained in the caption of Figure S22.

S4 Spectral Reflectance Curves for Six Species Across WV-3 Bands

The XAI permutation tests were useful for determining what the CNN models were relying on when learning relationships in the data to make skillful predictions. However, we found that examining the spectral reflectance curves of the six species was also helpful for understanding whether limitations in the WV-3 data were due to spatial or spectral resolution. We used the terra package (version 1.8.5) in R (version 4.4.2) to extract radiance values for all eight bands within each species polygon. We calculated the mean for each polygon, then used the boot package (version 1.3-31) to calculate the mean radiance for each species (across all polygons—the mean of means) with bootstrapped 95% confidence intervals. We used ggplot2 (version 3.5.1) to generate figures comparing all species reflectance curves on a single plot (Figure S25) and pairwise comparisons of species reflectance curves (Figures S26-S40).

Significant differences between species are difficult to discern when species curves are plotted together (Figure S25), but it is apparent that the coniferous species (subalpine fir - ABLA, Engelmann spruce - PIEN, and limber pine - PIFL) diverge from the deciduous species (glandular birch – BEGL and aspen – POTR) in the N-IR1 and N-IR2 bands, except for willow (*Salix spp*.), another deciduous species which overlaps with PIEN.

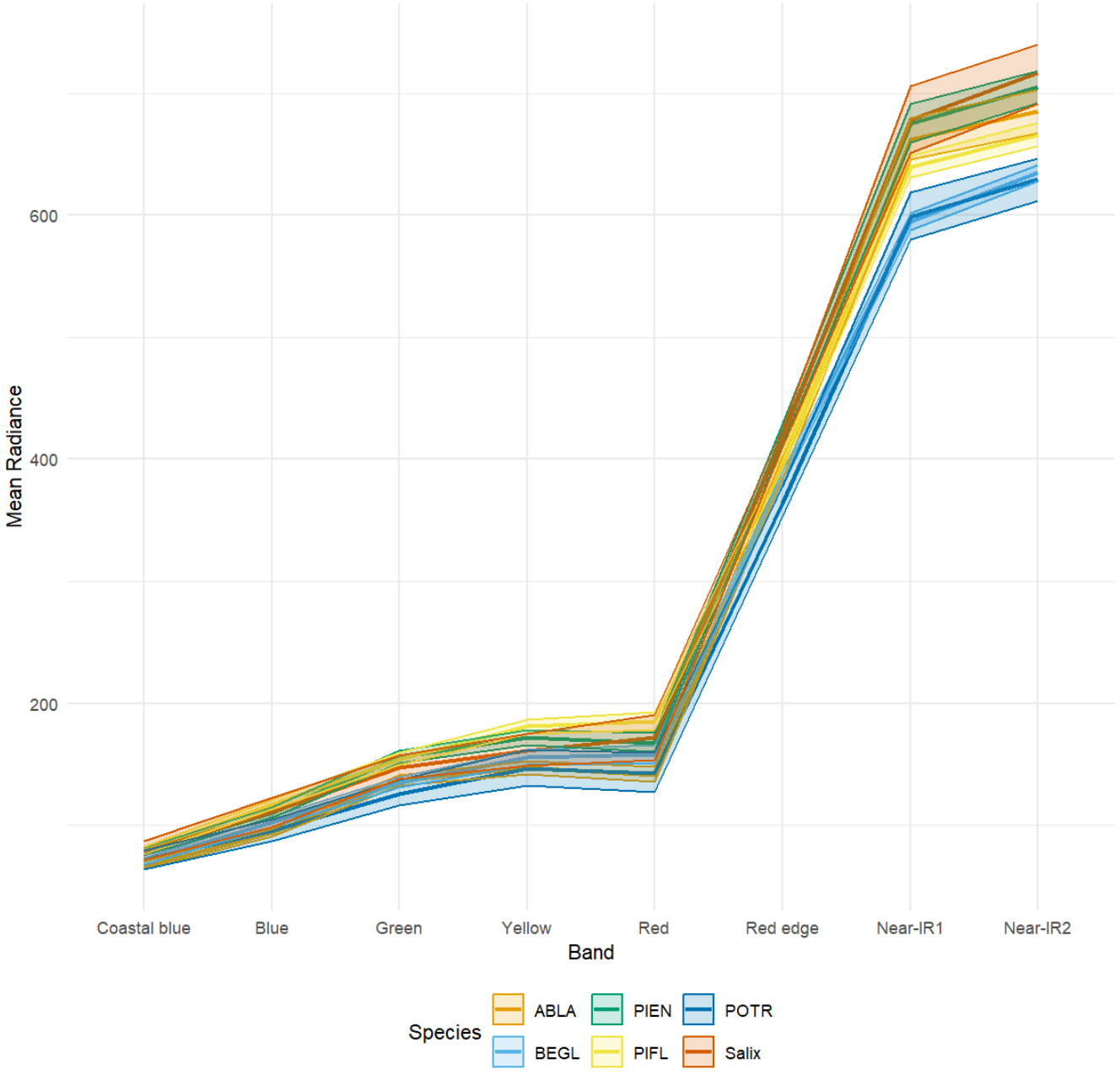


Figure S25. Spectral reflectance curves for all six species across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

ABLA and BEGL diverged significantly in the blue, red, N-IR1, and N-IR2 bands (Figure S26).

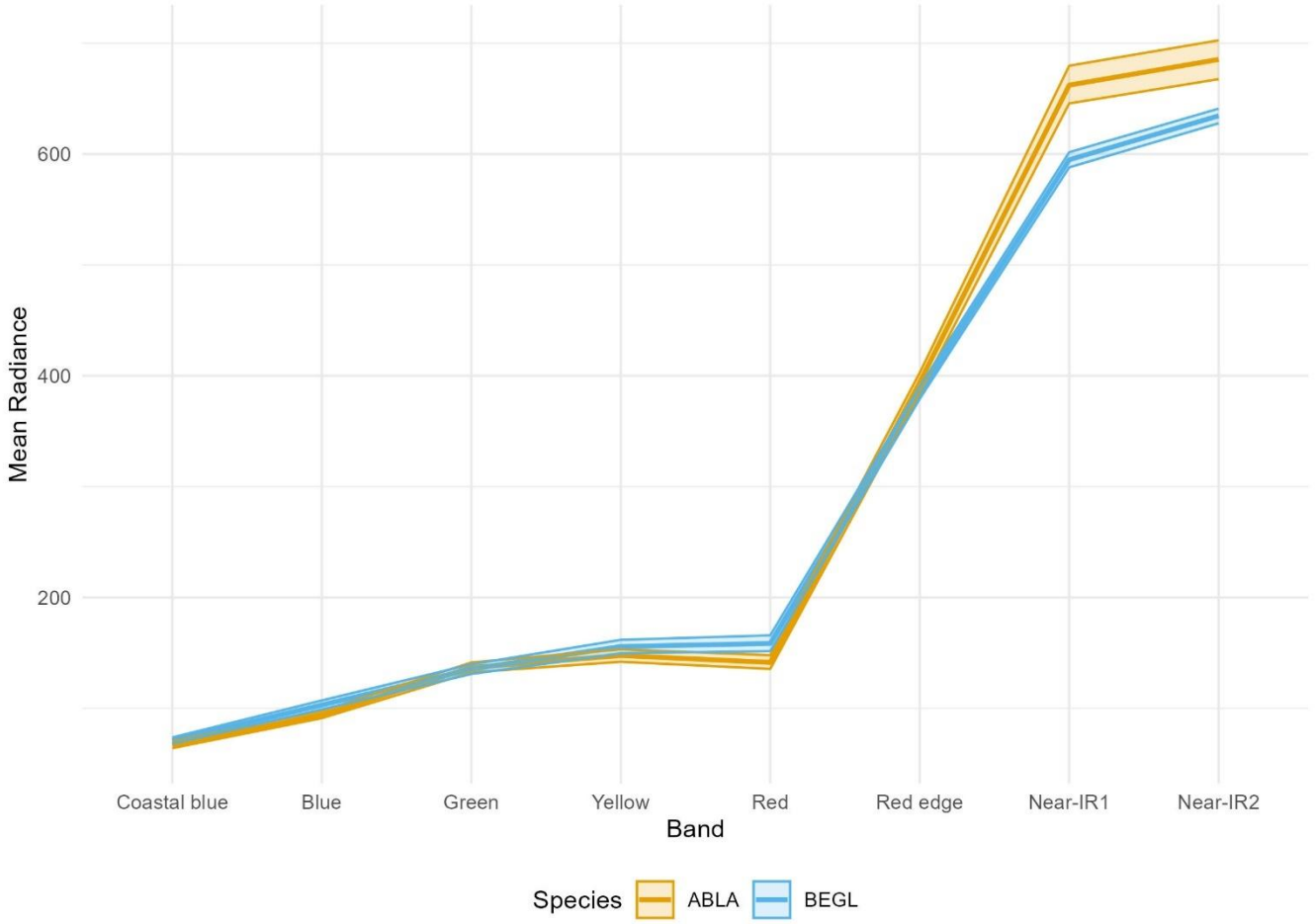


Figure S26. Spectral reflectance curves for subalpine fir (ABLA) and glandular birch (BEGL) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

ABLA and PIEN diverge significantly in the coastal blue, blue, green, yellow, red, and red edge bands (Figure S27).

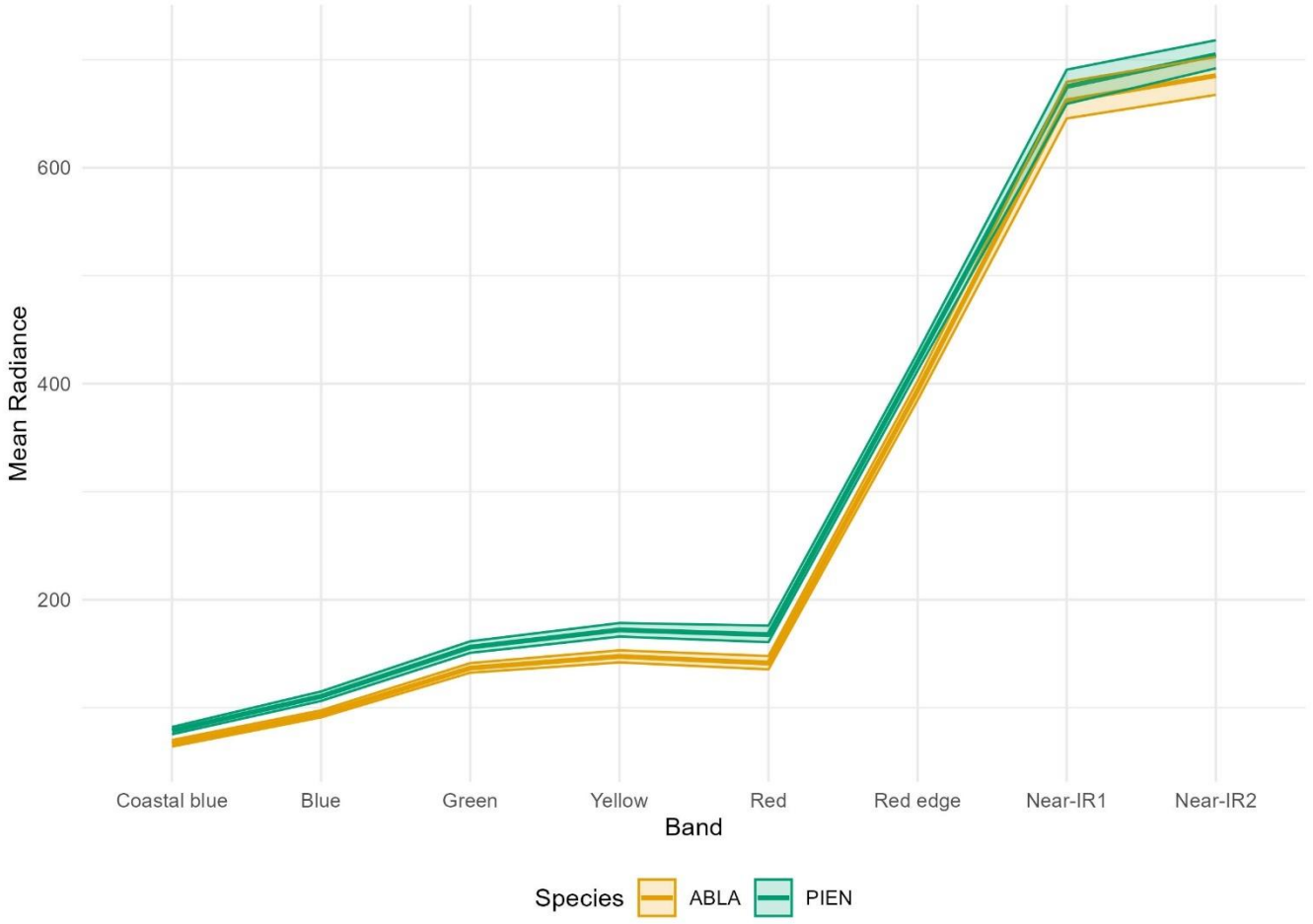


Figure S27. Spectral reflectance curves for subalpine fir (ABLA) and Engelmann spruce (PIEN) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

ABLA and PIFL diverge significantly in the coastal blue, blue, green, yellow, and red bands (Figure S28).

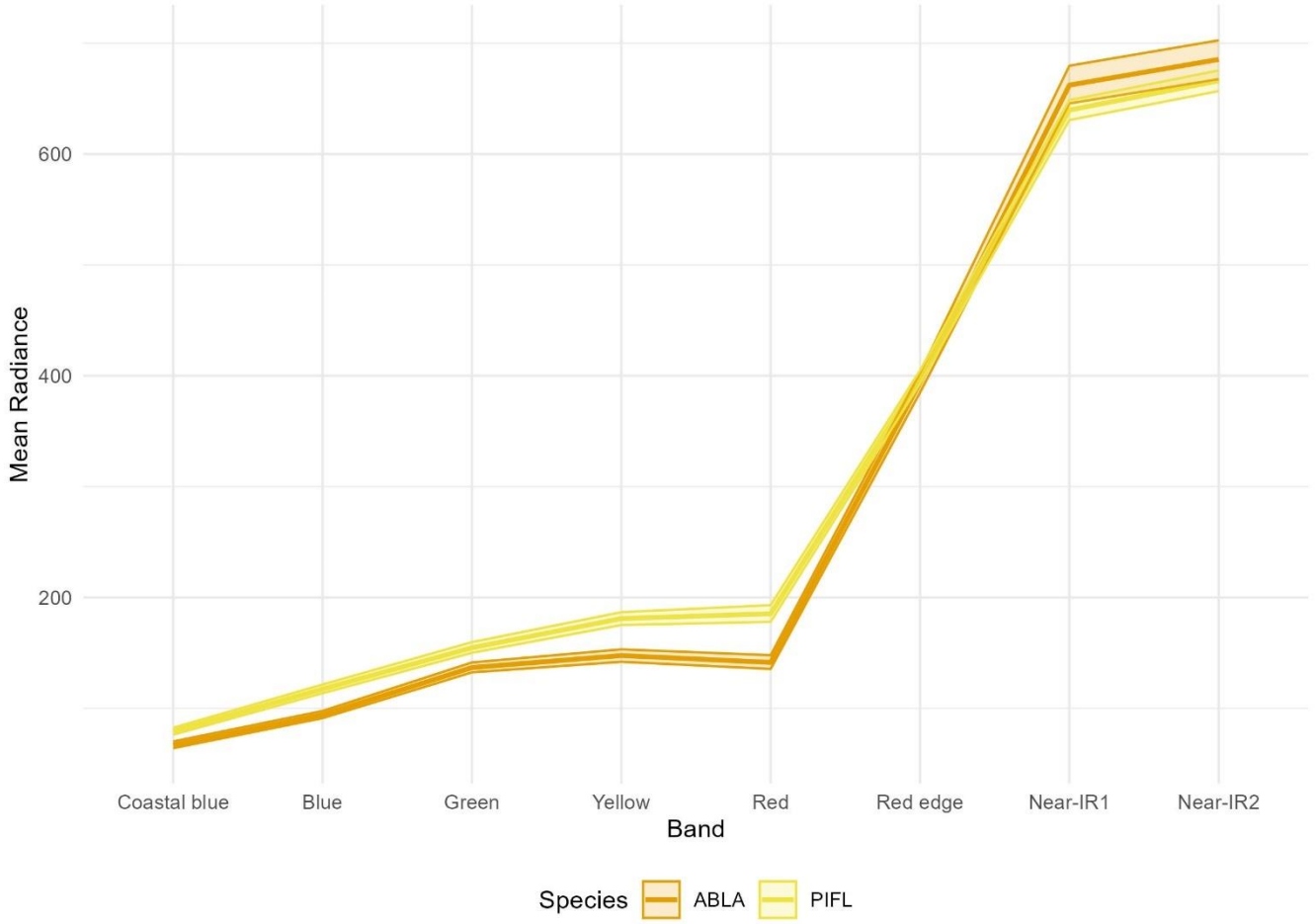


Figure S28. Spectral reflectance curves for subalpine fir (ABLA) and limber pine (PIFL) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

ABLA and POTR diverge significantly in the red edge, N-IR1, and NIR-2 bands (Figure S29).

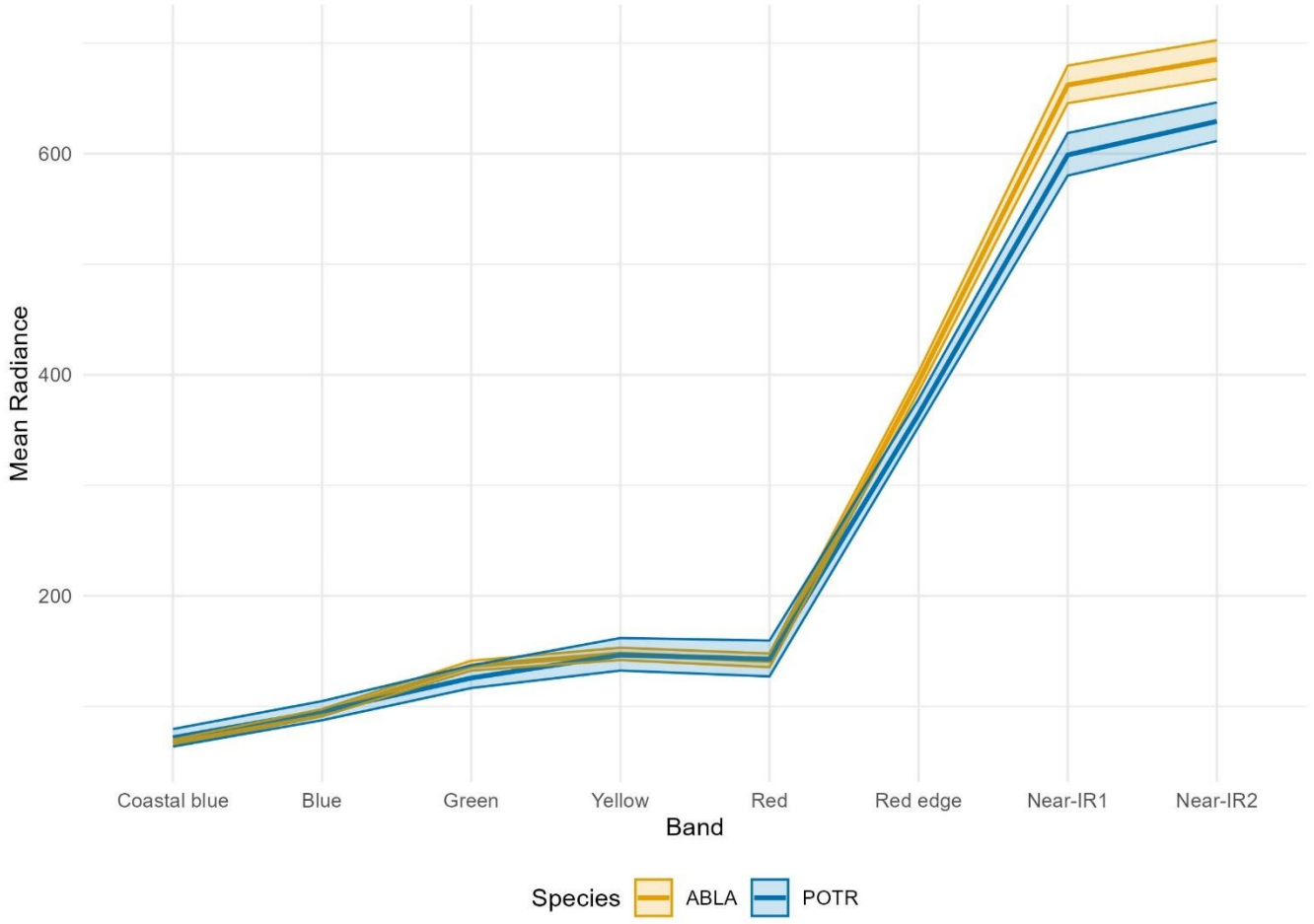
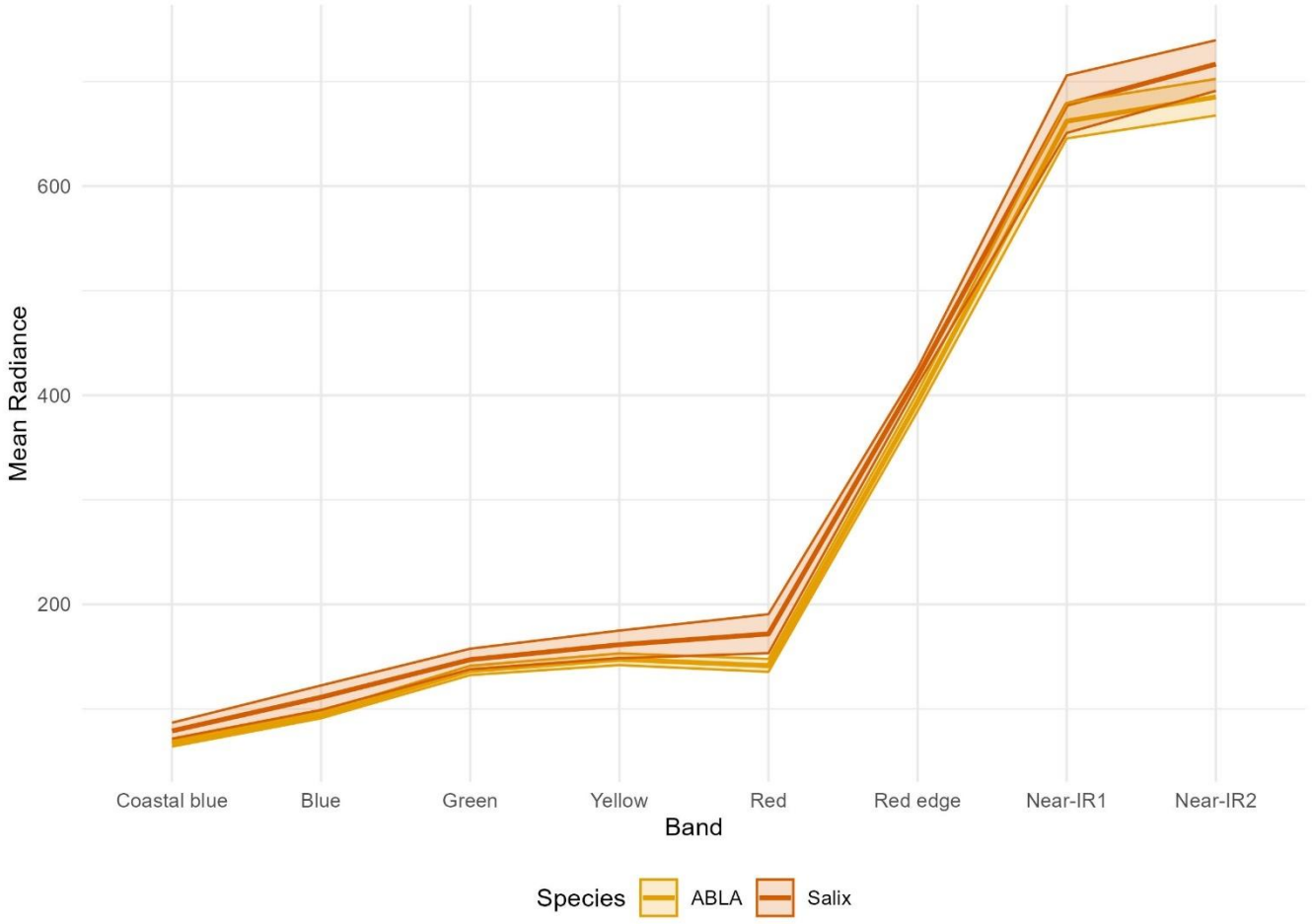


Figure S29. Spectral reflectance curves for subalpine fir (ABLA) and aspen (POTR) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

ABLA and *Salix* diverge significantly (if only slightly) in the coastal blue, blue, and red bands (Figure 30).



315

Figure S30. Spectral reflectance curves for subalpine fir (ABLA) and willow (*Salix*) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

BEGL and PIEN diverge significantly in the green, yellow, red edge, N-IR1, and N-IR2 bands (Figure S31).

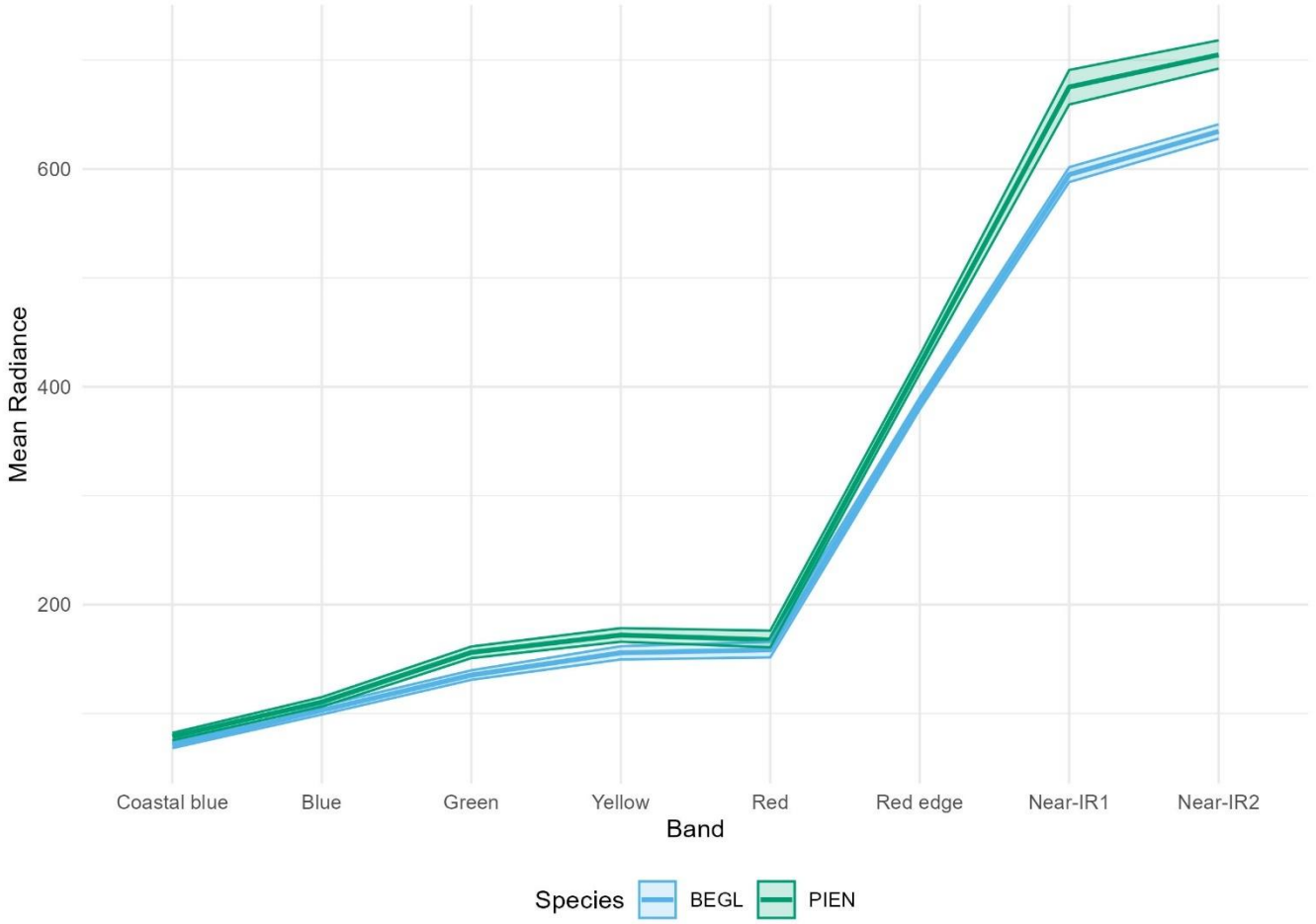


Figure S31. Spectral reflectance curves for glandular birch (BEGL) and Engelmann spruce (PIEN) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

BEGL and PIFL diverge significantly across the full spectrum (Figure S32).

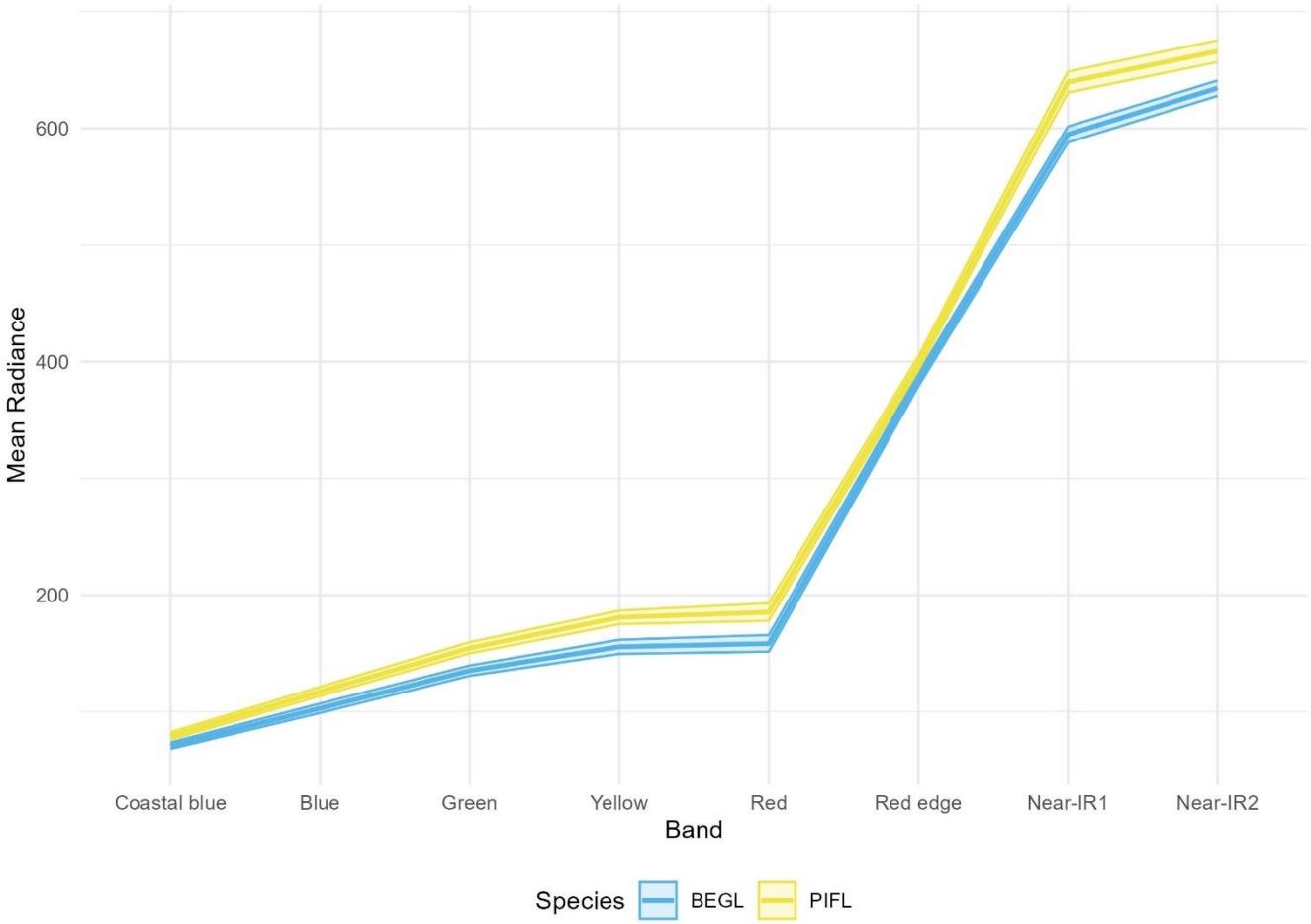


Figure S32. Spectral reflectance curves for glandular birch (BEGL) and limber pine (PIFL) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

BEGL and POTR may diverge slightly in the red edge band but are not distinguishable anywhere else along the spectrum (Figure S33).

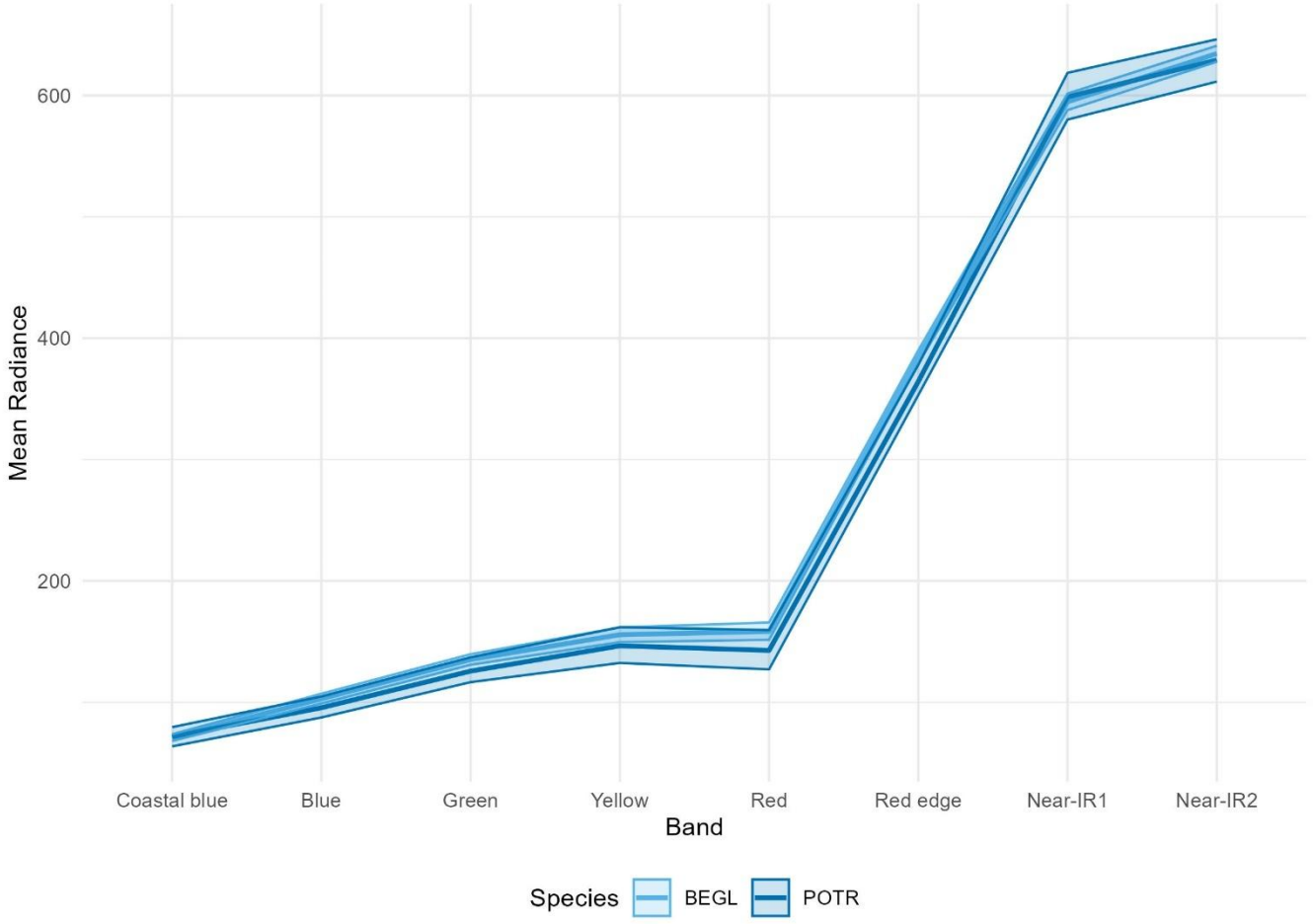
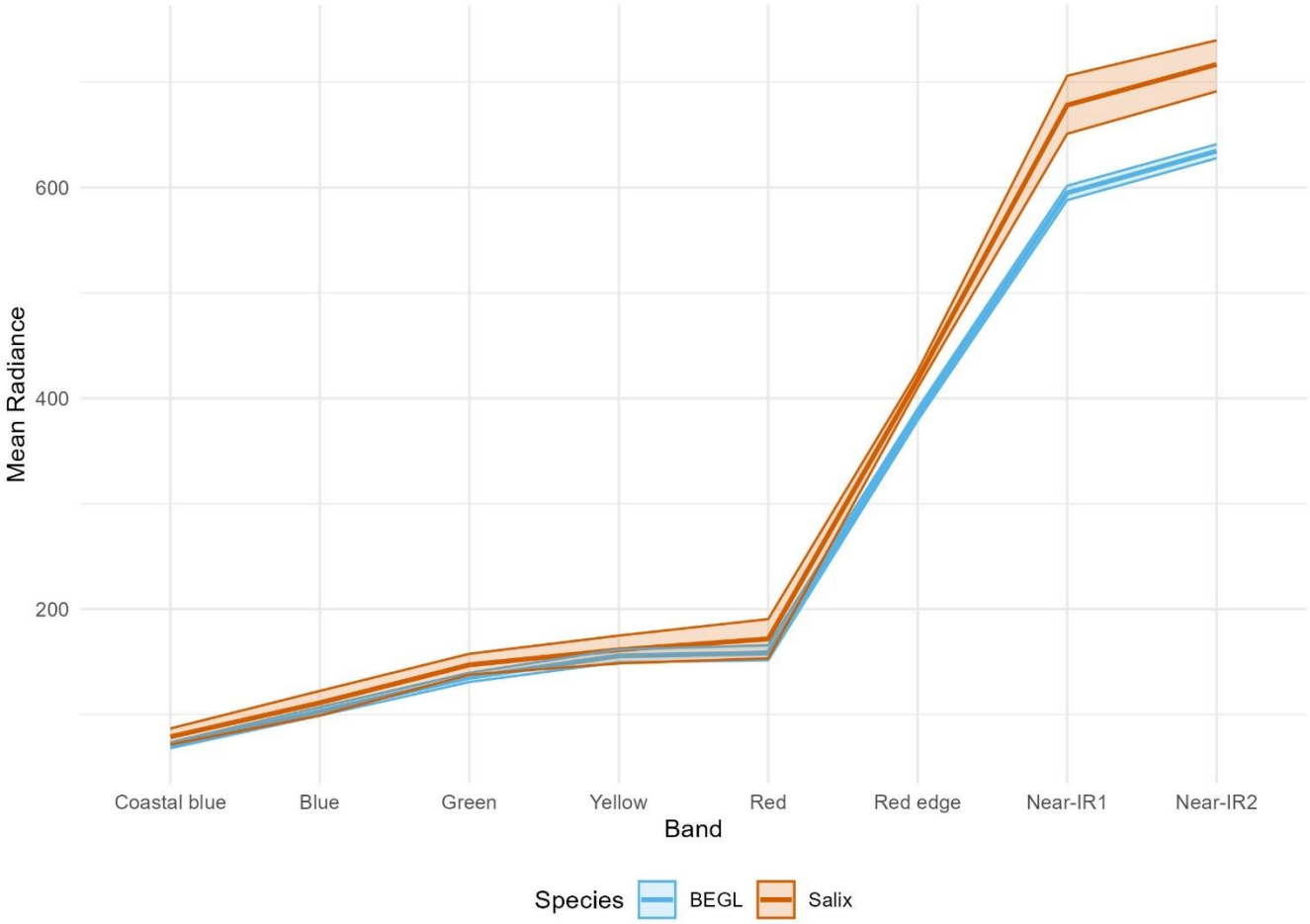


Figure S33. Spectral reflectance curves for glandular birch (BEGL) and aspen (POTR) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

BEGL and *Salix* diverge significantly only in the red edge, N-IR1, and N-IR2 bands (Figure S34).



340

Figure S34. Spectral reflectance curves for glandular birch (BEGL) and willow (*Salix*) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

PIEN and PIFL diverge significantly in the red, red edge, N-IR1, and N-IR2 bands (Figure S35).

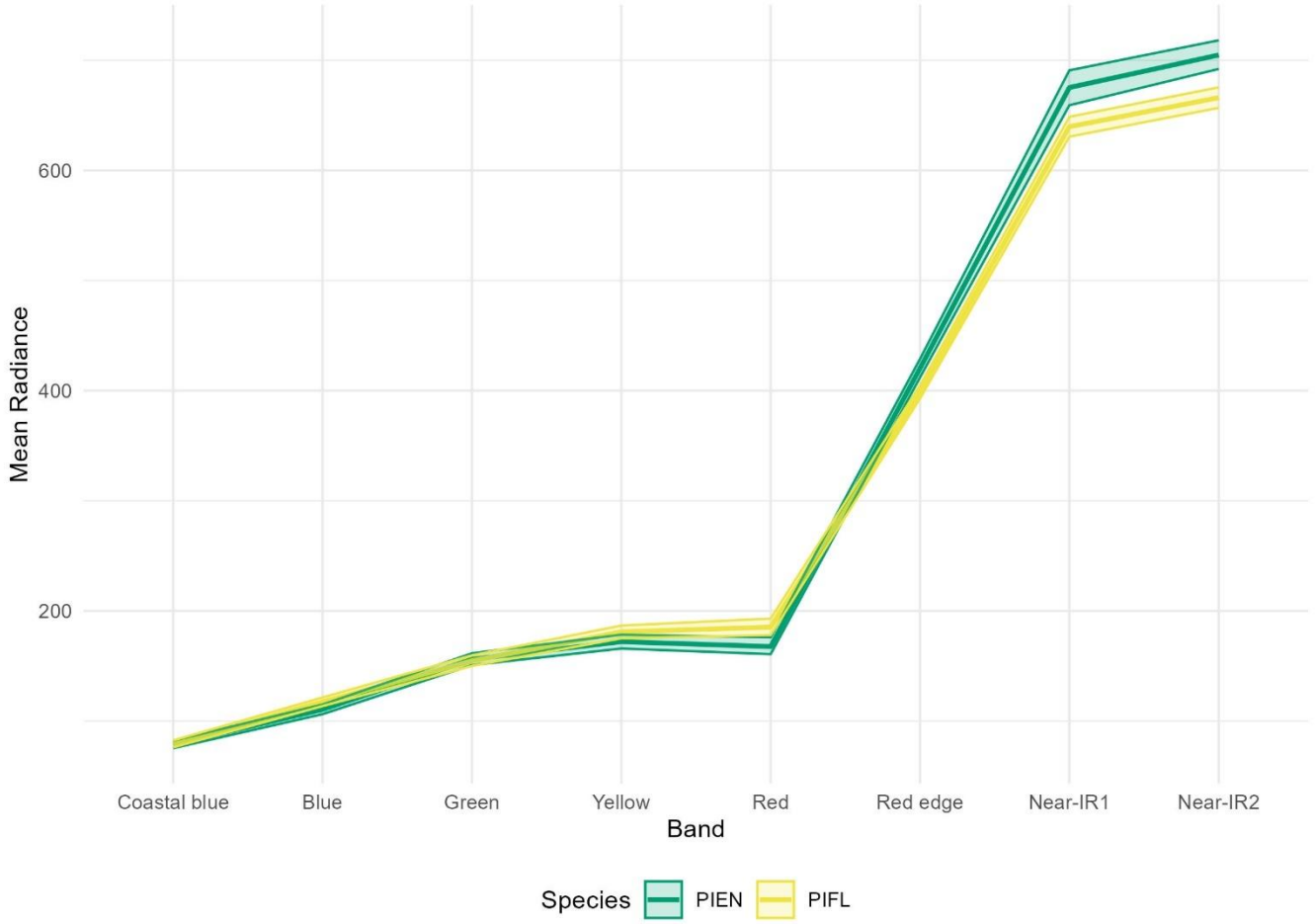


Figure S35. Spectral reflectance curves for Engelmann spruce (PIEN) and limber pine (PIFL) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

PIEN and POTR diverge significantly in every WV-3 band except for the costal band (Figure S36).

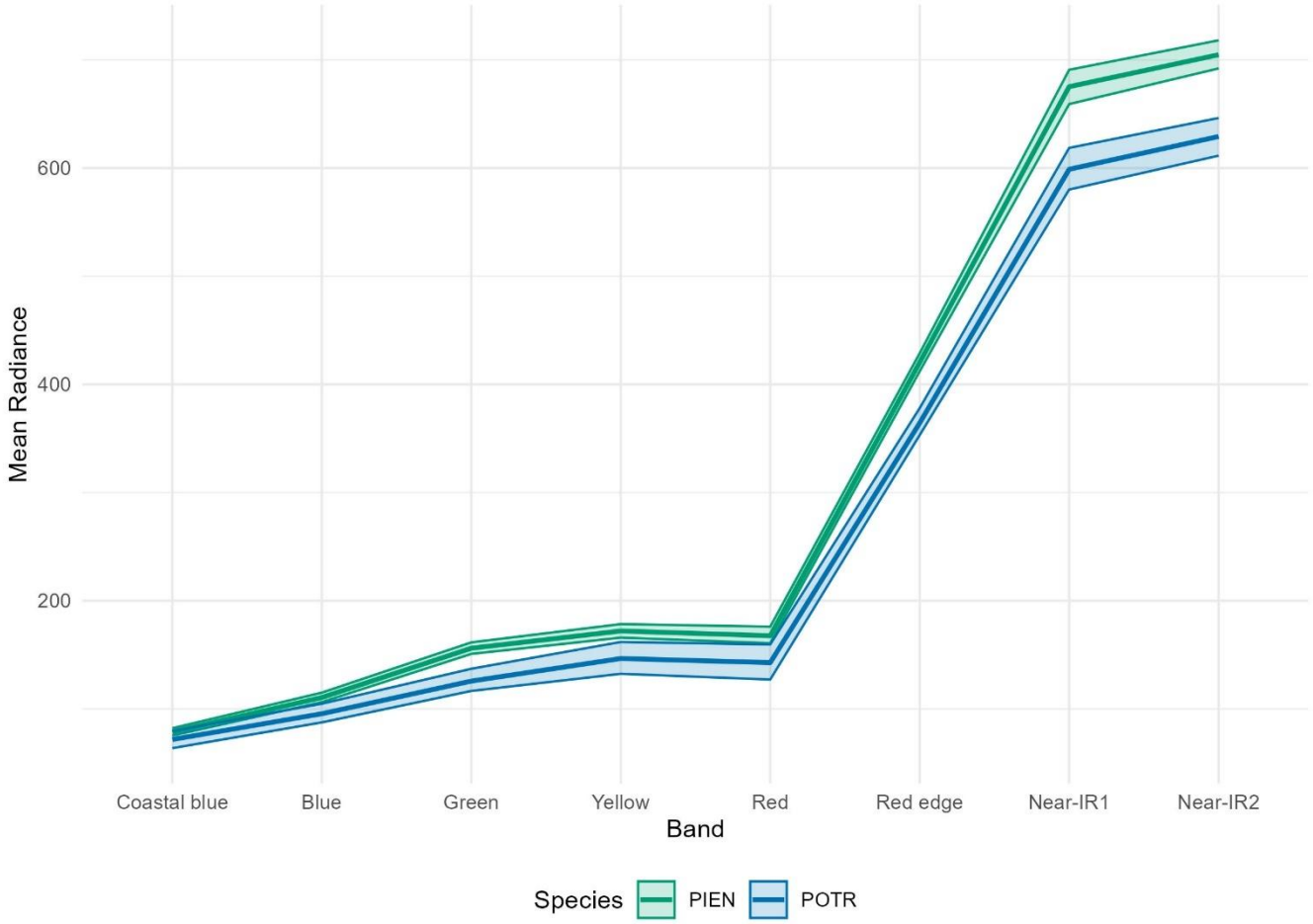


Figure S36. Spectral reflectance curves for Engelmann spruce (PIEN) and aspen (POTR) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

PIEN and *Salix* do not diverge significantly in any of the WV-3 bands (Figure 37).

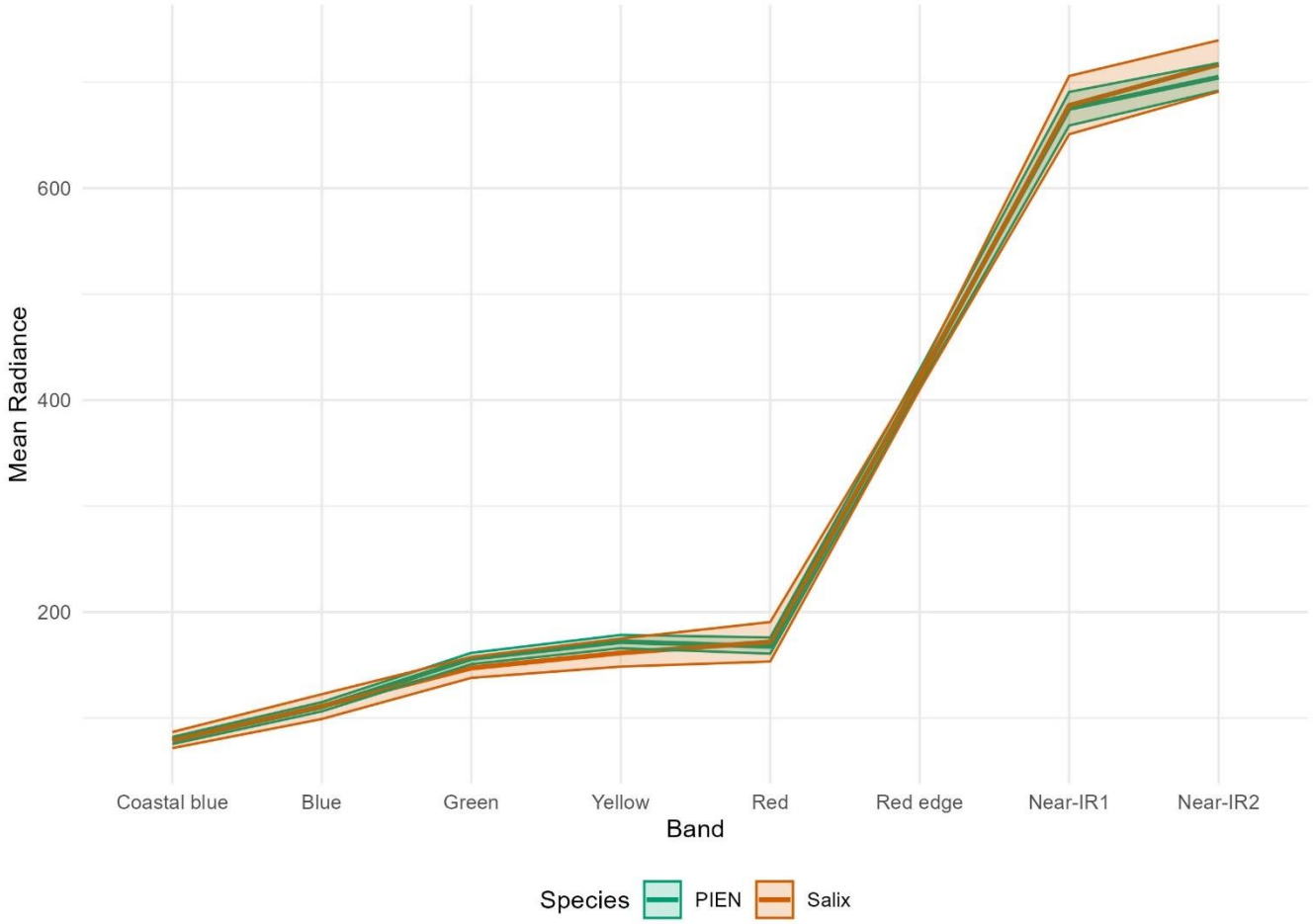


Figure S37. Spectral reflectance curves for Engelmann spruce (PIEN) and willow (*Salix*) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

PIFL and POTR diverge significantly in every WV-3 band except for the coastal band (Figure S38).

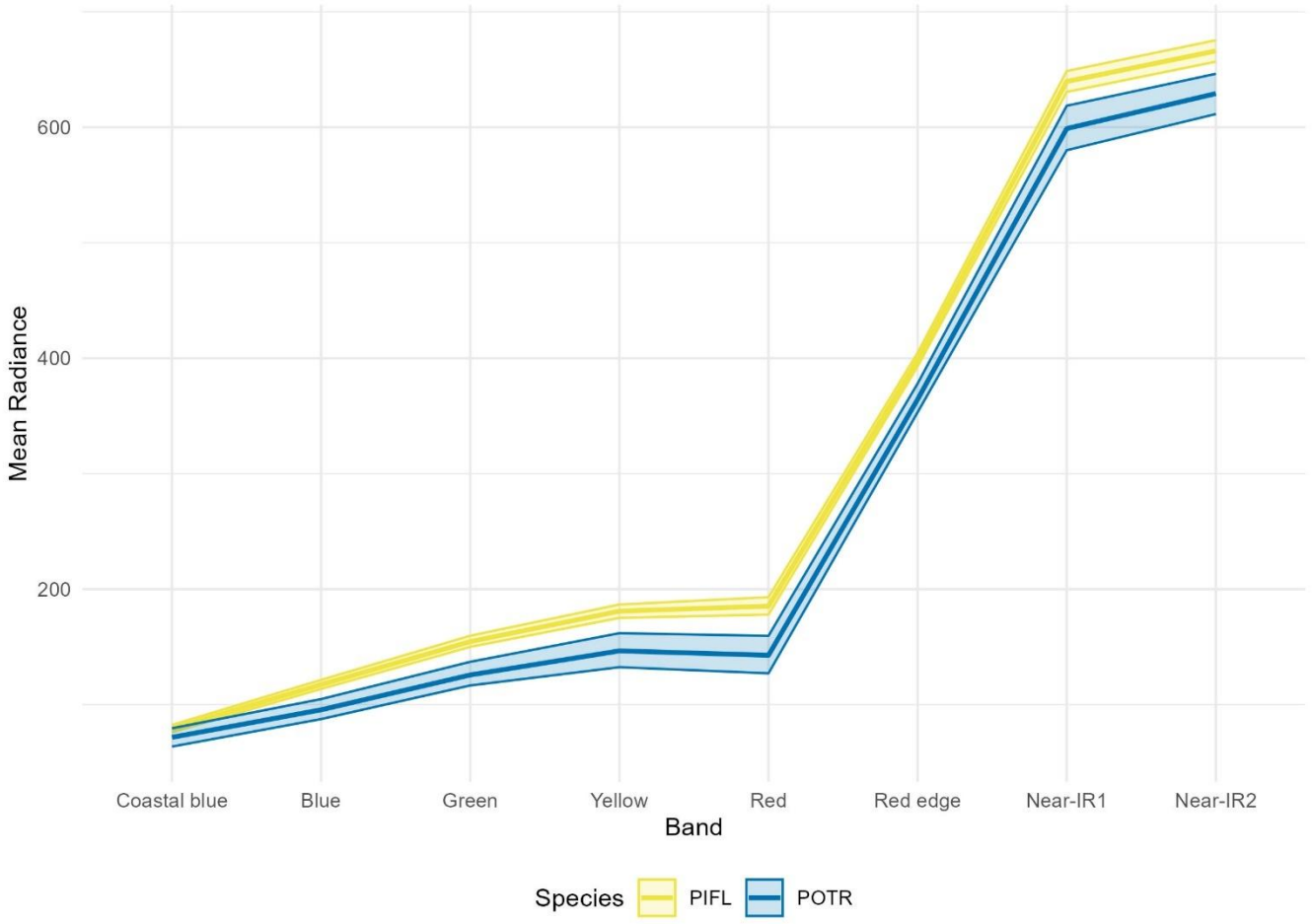


Figure S38. Spectral reflectance curves for limber pine (PIFL) and aspen (POTR) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

PIFL and *Salix* diverge significantly only in the N-IR1 and N-IR2 bands (Figure S39).

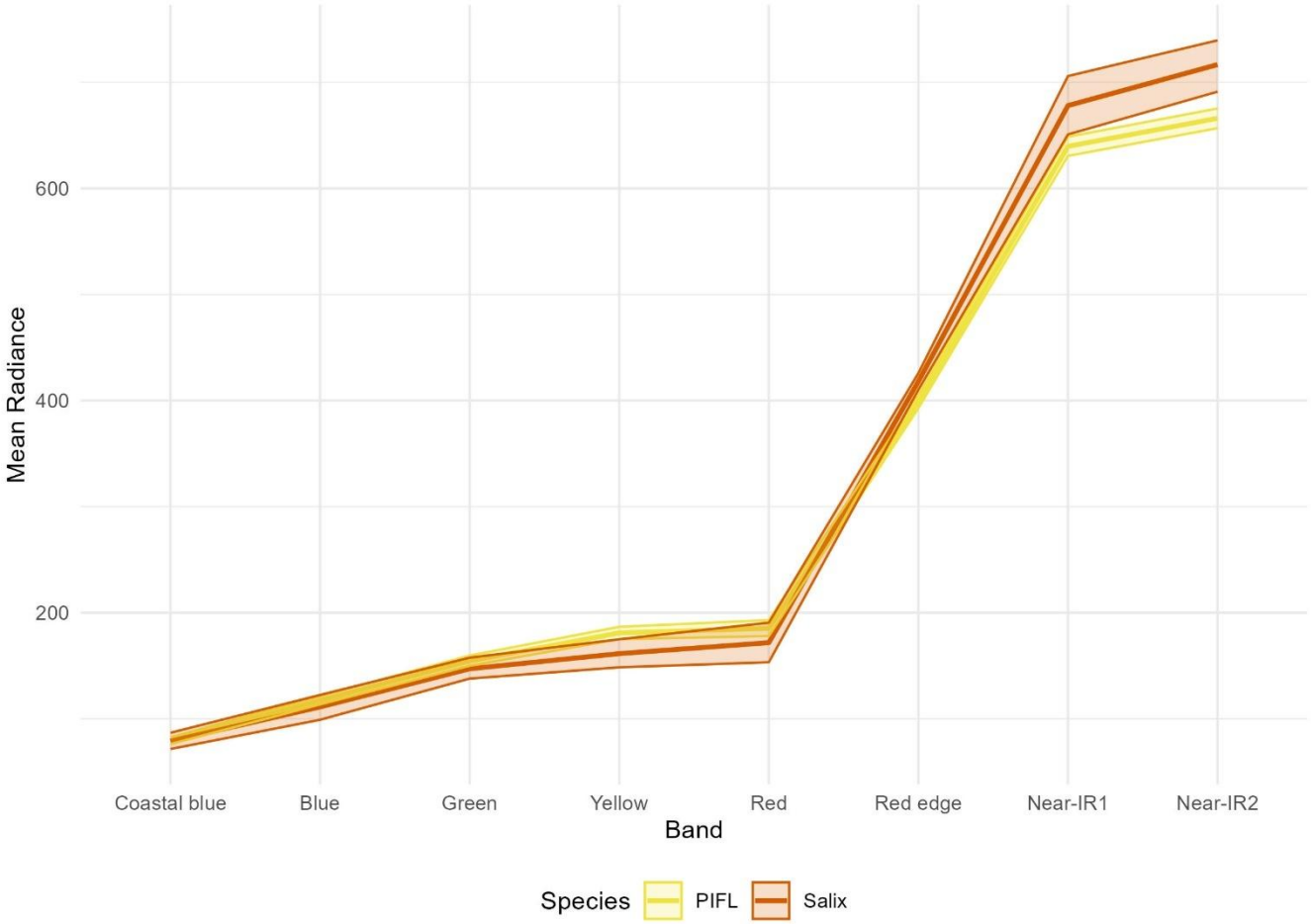


Figure S39. Spectral reflectance curves for limber pine (PIFL) and willow (*Salix*) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

375 POTR and *Salix* diverge significantly in the red edge, N-IR1, and N-IR2 bands (Figure S40).

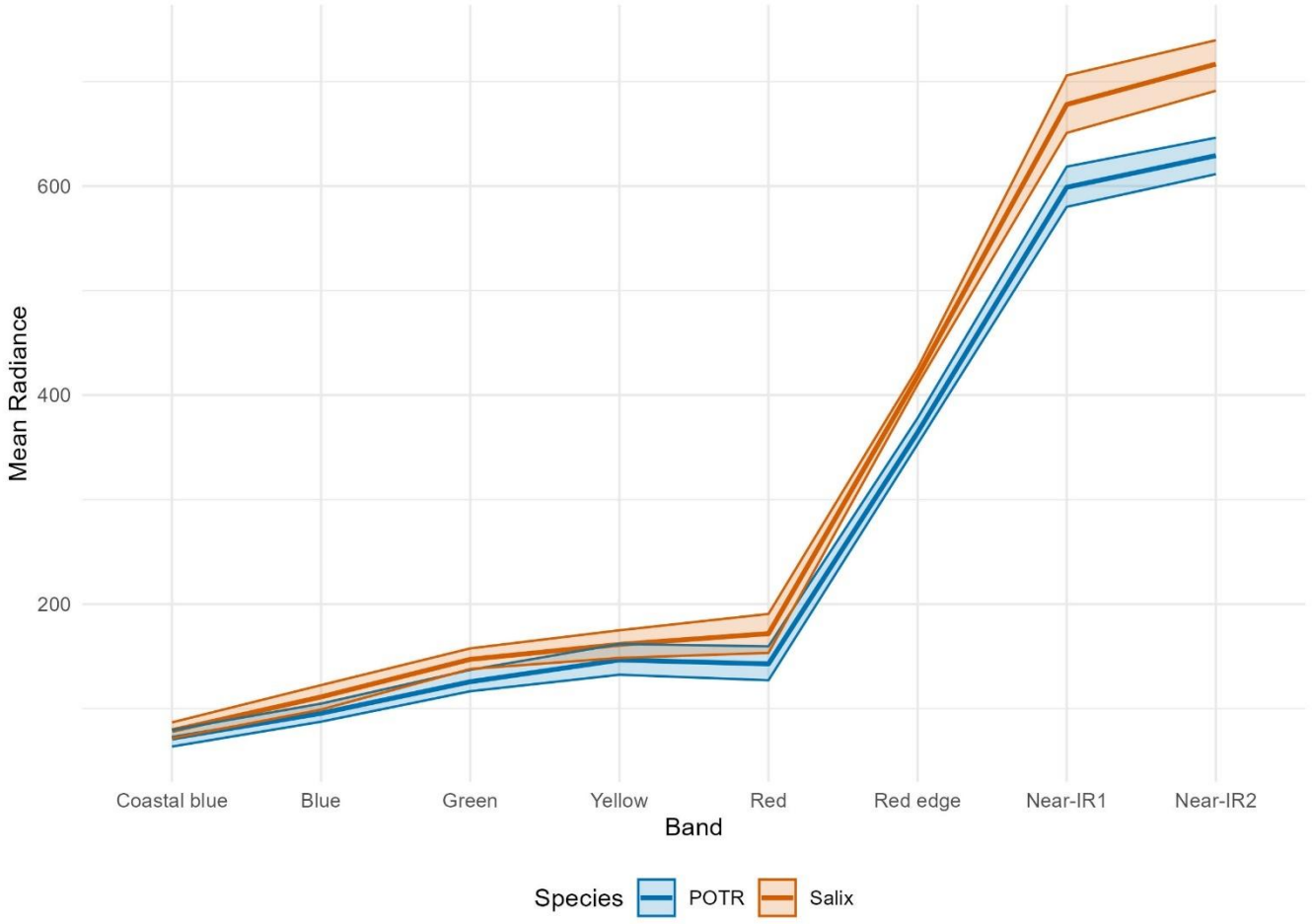


Figure S40. Spectral reflectance curves for aspen (POTR) and willow (*Salix*) across the eight WV-3 bands. Each line shows mean radiance for a different species across all polygons, with shaded 95% confidence intervals. Units of radiance are in W m^{-2} sr^{-1} μm^{-1}.

Even with eight-band multispectral data (vs. hyperspectral data), the six species do diverge significantly from one another in
380 different regions of the electromagnetic spectrum, except for Engelmann spruce and willow, which overlap in all eight bands. Table S1 provides a summary of bands where 95% confidence intervals of mean radiance (across species polygons) do not overlap, indicating a significant difference between measures of species reflectance in those bands.

Table S1. WV-3 bands where pairs of species diverge significantly—where 95% confidence intervals for mean radiance (across species polygons) do not overlap.

Species Combinations	WV-3 Bands where Species Diverge Significantly
ABLA & BEGL	Blue, Red, N-IR1, N-IR2
ABLA & PIEN	Coastal blue, Blue, Green, Yellow, Red, Red edge
ABLA & PIFL	Coastal blue, Blue, Green Yellow, Red
ABLA & POTR	Red edge, N-IR1, N-IR2
ABLA & *Salix*	Coastal blue, Blue, Red
BEGL & PIEN	Green, Yellow, Red edge, N-IR1, N-IR2
BEGL & PIFL	All bands
BEGL & POTR	Red edge
BEGL & *Salix*	Red edge, N-IR1, N-IR2
PIEN & PIFL	Red, Red edge, N-IR1, N-IR2
PIEN & POTR	Blue, Green, Yellow, Red, Red edge, N-IR1, N-IR2
PIEN & *Salix*	No bands
PIFL & POTR	Blue, Green, Yellow, Red, Red edge, N-IR1, N-IR2
PIFL & *Salix*	N-IR1, N-IR2
POTR & *Salix*	Red edge, N-IR1, N-IR2

S4 Permutation Test Results for Species-Specific Model Performance

Each subsection below shows the results from a set of permutation tests, used to evaluate predictor importance for accurate predictions of specific classes in the six-class, four-class, and two-class models. For permutation tests shown in the main body, the Gerrity score was used as the evaluation metric, which is designed for multi-class classification and therefore considers discrimination among all K classes. To evaluate model performance for each class, we used a metric that considers only discrimination between each "one" species and all others: the area under the receiver operating characteristic (ROC) curve (Figure S41).

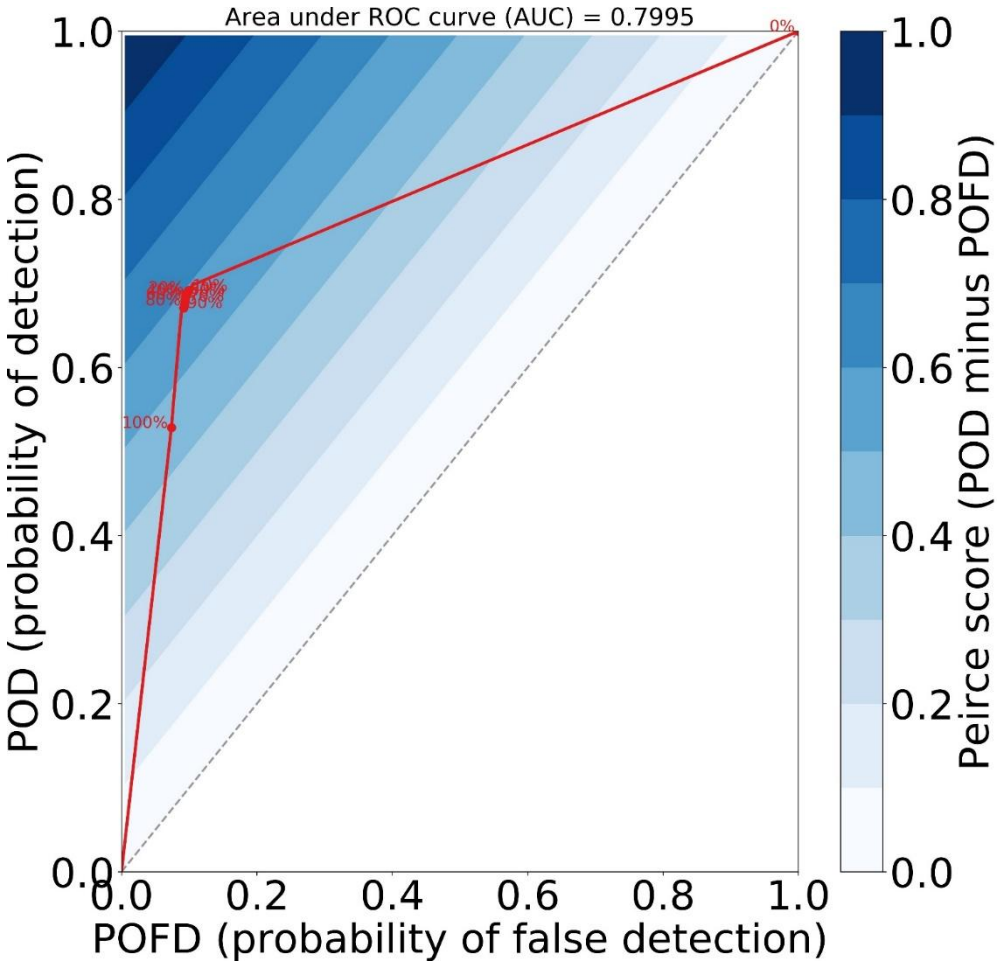


Figure S41. ROC curve for the 2-class model (PIFL vs. Other). For this ROC curve, the positive (negative) class is chosen to be PIFL (Other). Hence, the POD is # *correctly identified PIFL*/# *total PIFL*, while the POFD is # *Other falsely identified as PIFL*/# *total Other*. Each point in the ROC curve corresponds to a different probability threshold p* (labelled at 10% intervals), such that PIFL probabilities ≥ p* are "yes" predictions and PIFL probabilities < p* are "no" predictions. For binary (2-class) classification, the Peirce score is POD minus POFD, allowing it to be contoured on the axes of the ROC curve.

The ROC curve is designed for binary (yes/no) classification. (Thus, for the two-class model, a ROC curve can be computed trivially.) ROC curves plot the relationship between the true positive rate (sensitivity) and the false positive rate (specificity). The area under the ROC curve (AUC) is a measure of model performance. A random model tends to follow the one-to-one line across the plot (TPR = FPR for all thresholds), yielding an AUC of 0.5. For a perfect model, TPR = 1.0 and FPR = 0.0 for all thresholds, yielding an AUC of 1.0. A good model approaches this, and so the AUC is high.

For a K-class model with K > 2, the problem must be turned into K "one vs. all" binary problems; then a ROC curve must be computed for each binary problem. For example, the four-class model discriminates among four species: PIFL, ABLA, PIEN, and Other. Four ROC curves can be created for this model: one evaluating binary classification for PIFL (where the

"yes" event is PIFL and the "no" event is all non-PIFL species—ABLA, PIEN, or Other), one for ABLA vs. all, one for PIEN vs. all, and one for Other vs. all.

More specifically, the ROC curve is designed for *deterministic* binary classification (a model that outputs "yeses" and "nos"), but our models output probabilities. To turn these probabilities into "yeses" and "nos," we apply 1001 probability thresholds: 0.000, 0.001, 0.002, ..., 0.999, 1.000. For each threshold p*, probabilities >= p* become "yes" predictions and probabilities < p* become "no" predictions. The ROC curve plots the true-positive rate (TPR; number of true positives / [number of true positives + number of false negatives]) vs. the false-positive rate (FPR; number of false positives / [number of false positives + number of true negatives]) for every one of these thresholds. The use of multiple thresholds allows for a continuous curve in TPR/FPR space.

S3.1 Six-Class Model Permutation Tests

Figure 9 of the main body shows results from all four versions of the permutation test (single-pass forward, multi-pass forward, single-pass backward, and multi-pass backward), for the six-class model, using the Gerrity score to consider discrimination among all species. Figures S42-S47 are analogous; each figure shows all four versions of the permutation test, for the six-class model, but now using the AUC metric to consider discrimination of only one species from others. Thus, results presented in the main body illuminate the most important predictors for differentiating among all classes, while results presented herein illuminate the most important predictors for discriminating just one class from the others.

Figure S42. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying limber pine (PIFL) in the six-class model. Predictors in bold have a significant effect on model performance when permuted, according to a 95% confidence interval over 100 random perturbations of the given predictor. Within each panel, predictor importance decreases from top to bottom, so the most important predictors are at the top. Model performance here was evaluated by the area under the receiver operating characteristic curve (AUC) with respect to distinguishing limber pine from other species.

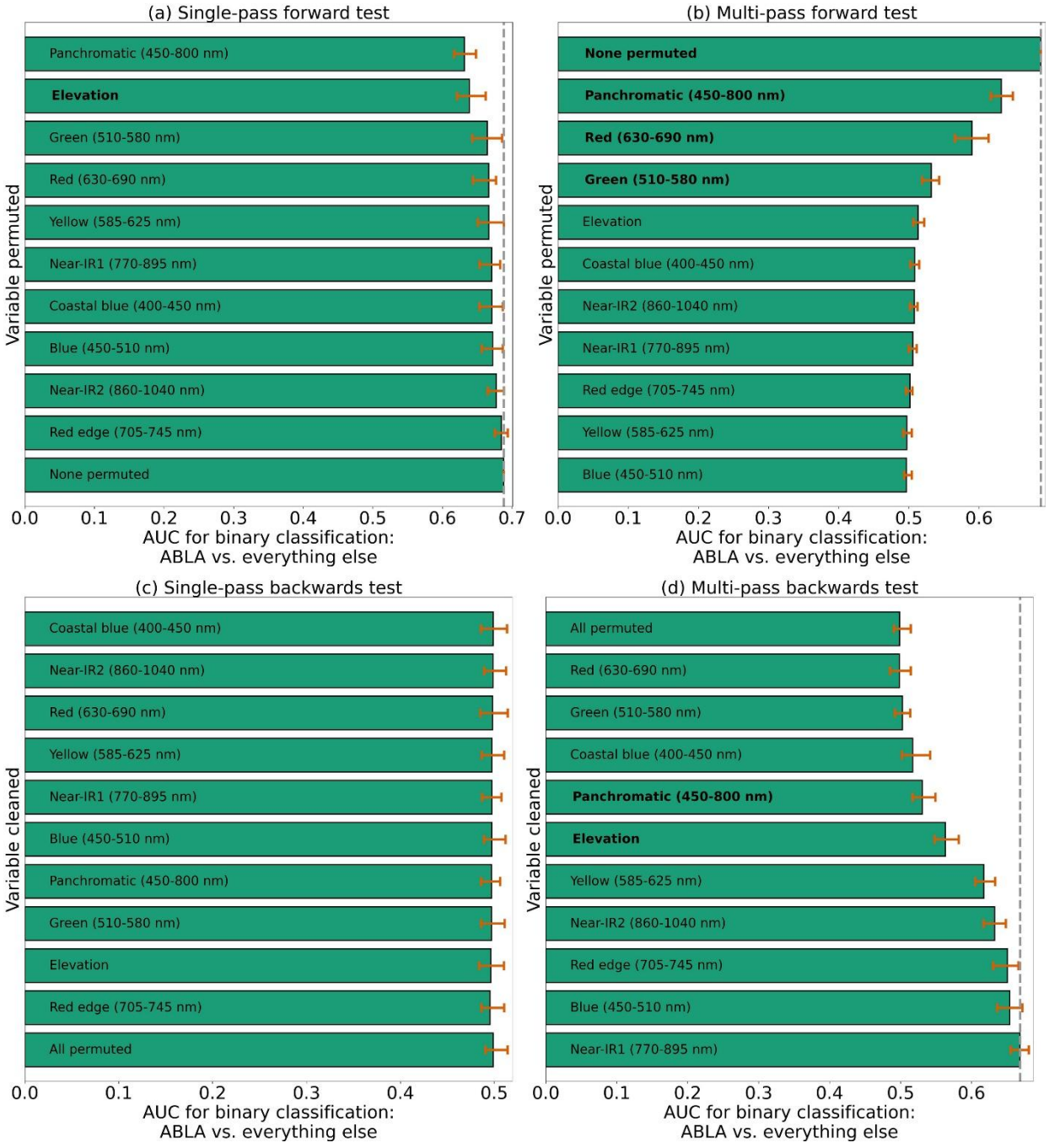


Figure S43. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying subalpine fir (ABLA) in the six-class model. See the caption for Figure S42 for more details.

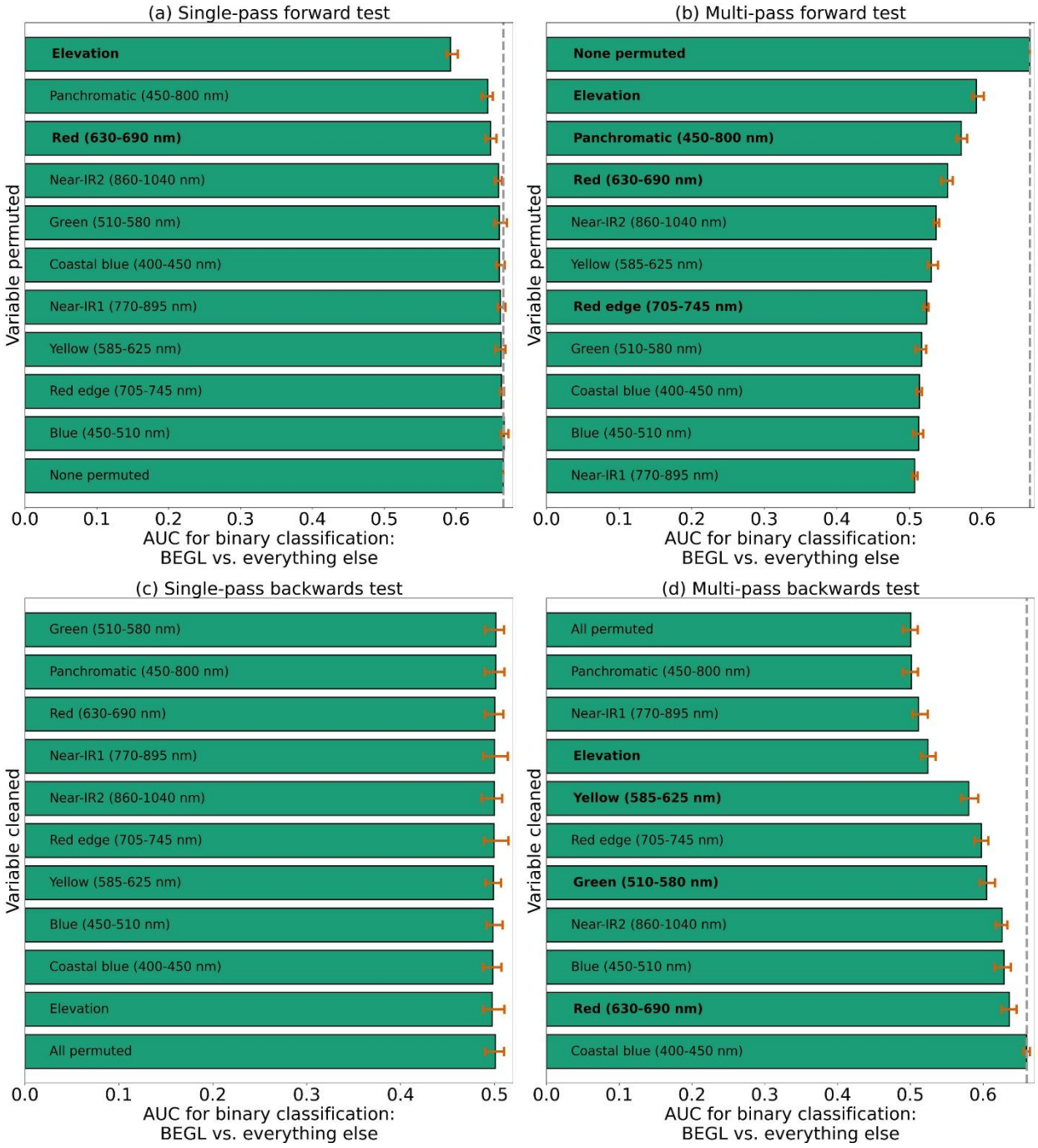


Figure S44. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying glandular birch (BEGL) in the six-class model. See the caption for Figure S42 for more details.

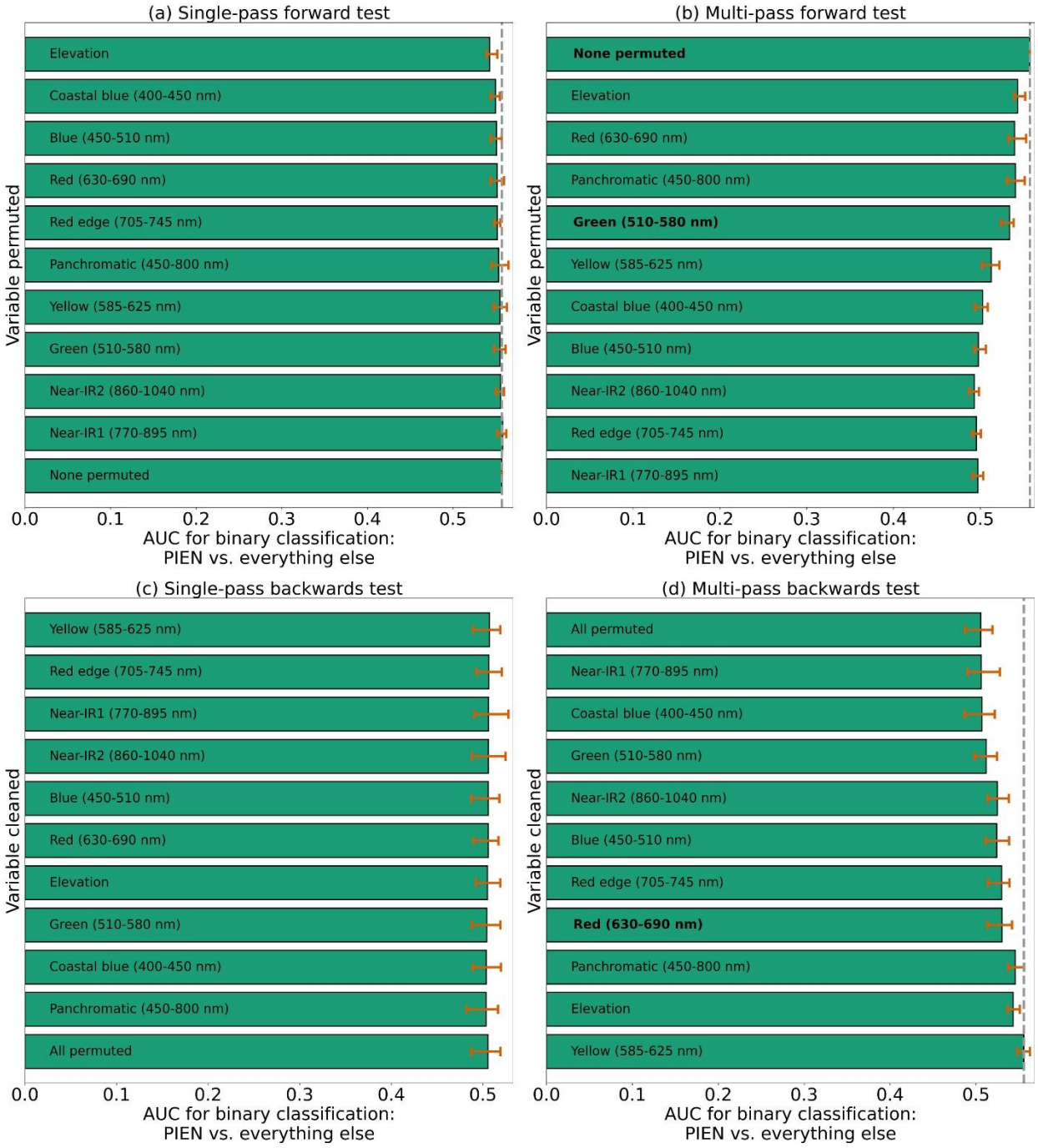
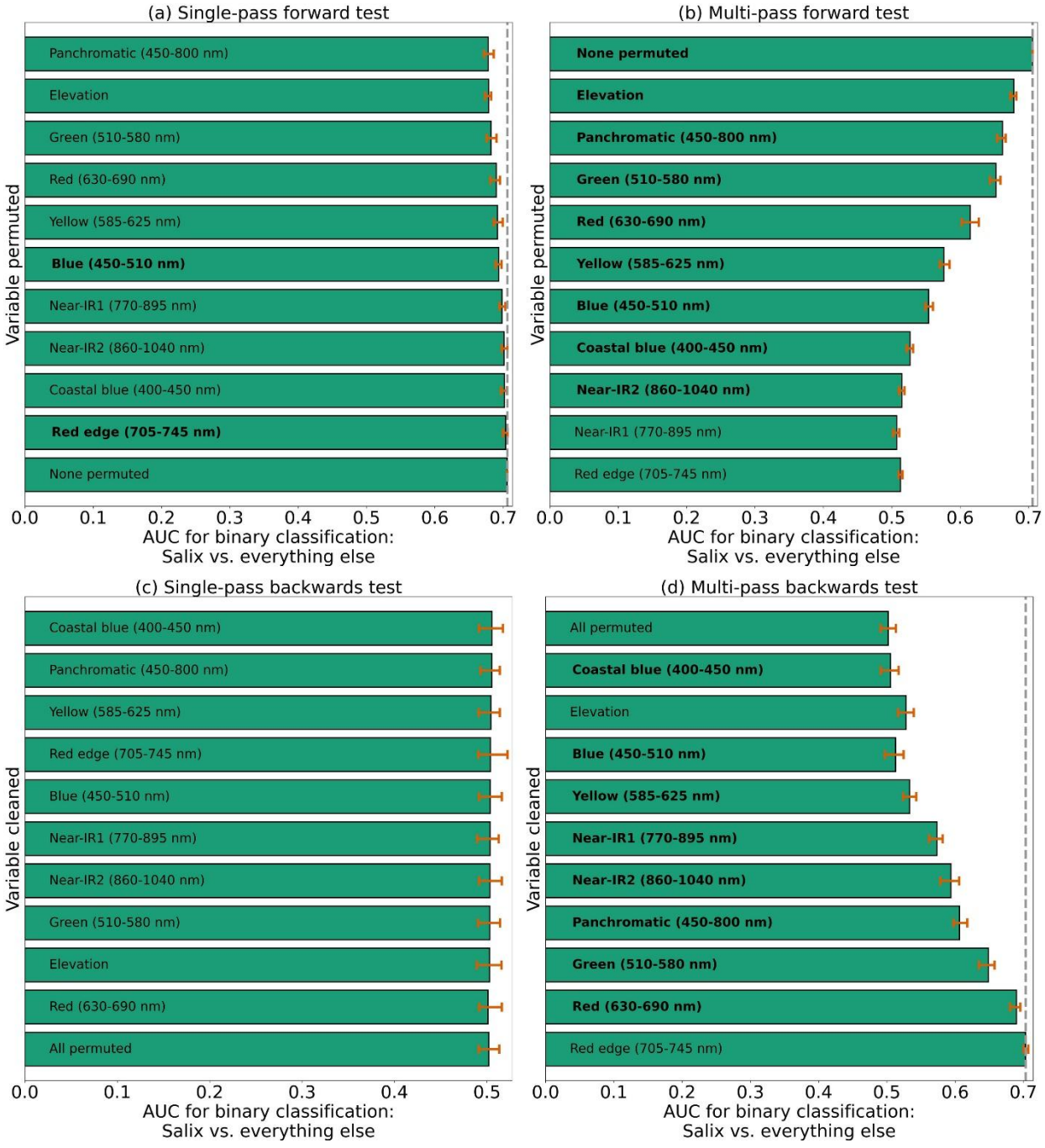


Figure S45. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying Engelmann spruce (PIEN) in the six-class model. See the caption for Figure S42 for more details.

Figure S46. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying aspen (POTR) in the six-class model. See the caption for Figure S42 for more details.



445

Figure S47. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying willow (*Salix*) in the six-class model. See the caption for Figure S42 for more details.

S3.2 Four-Class Model Permutation Tests

Figure 11 of the main body shows results of the permutation test, for the four-class model, using the Gerrity score to consider discrimination among all species. Figures S48-S51 are analogous but using the AUC metric to consider discrimination of only one species from others.

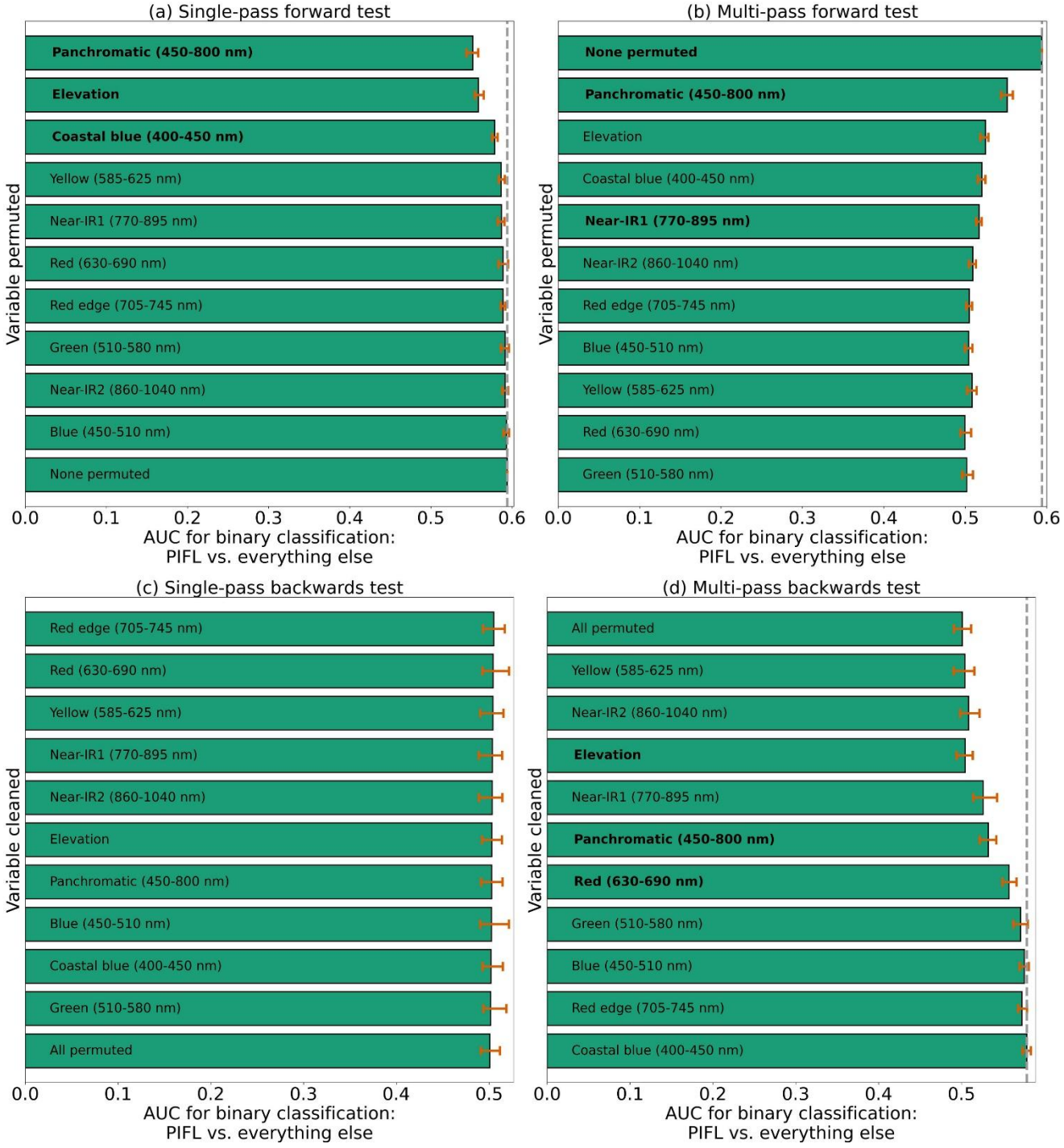


Figure S48. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying PIFL in the four-class model. See the caption for Figure S42 for more details.

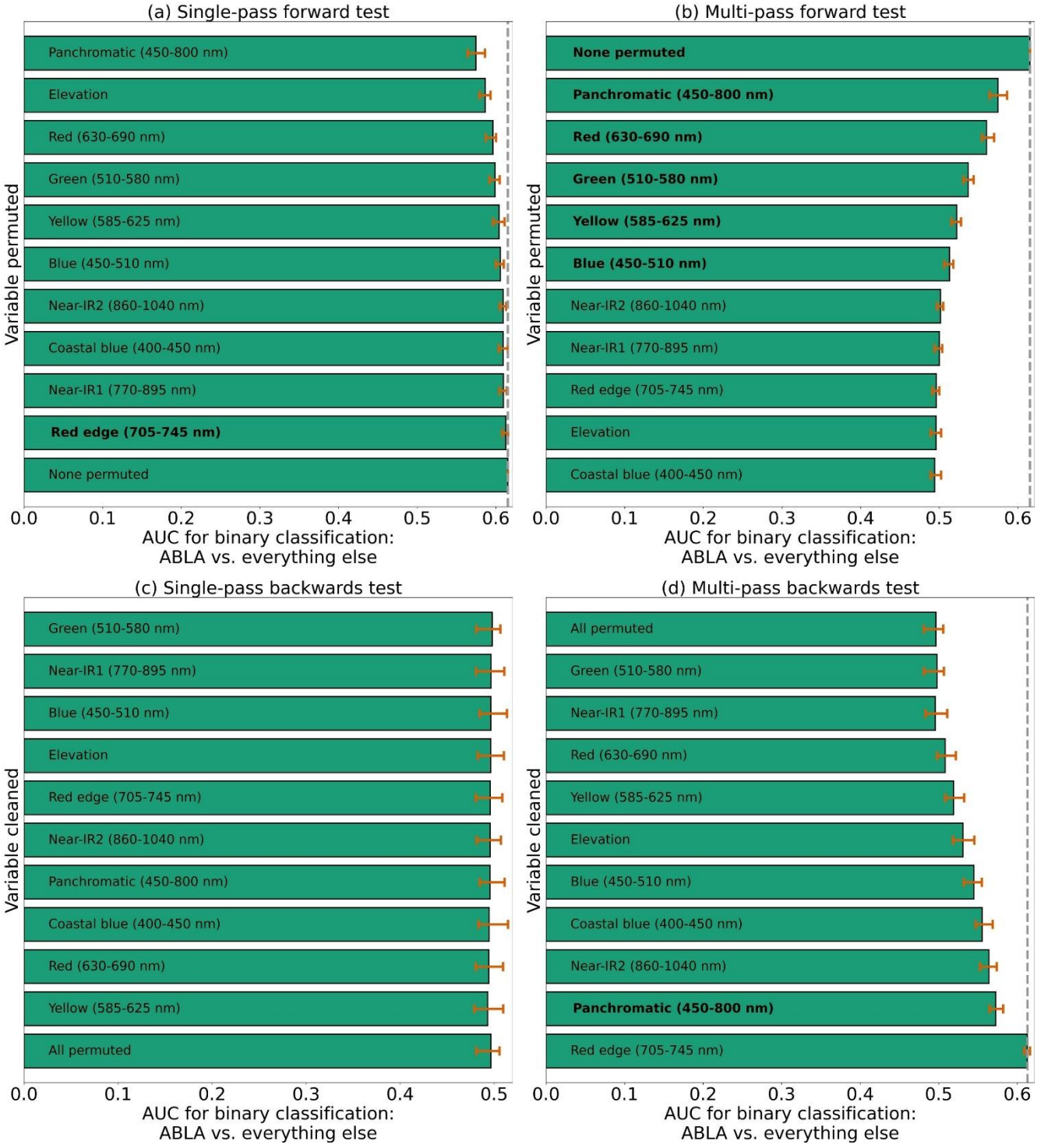


Figure S49. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying ABLA in the four-class model. See the caption for Figure S42 for more details.

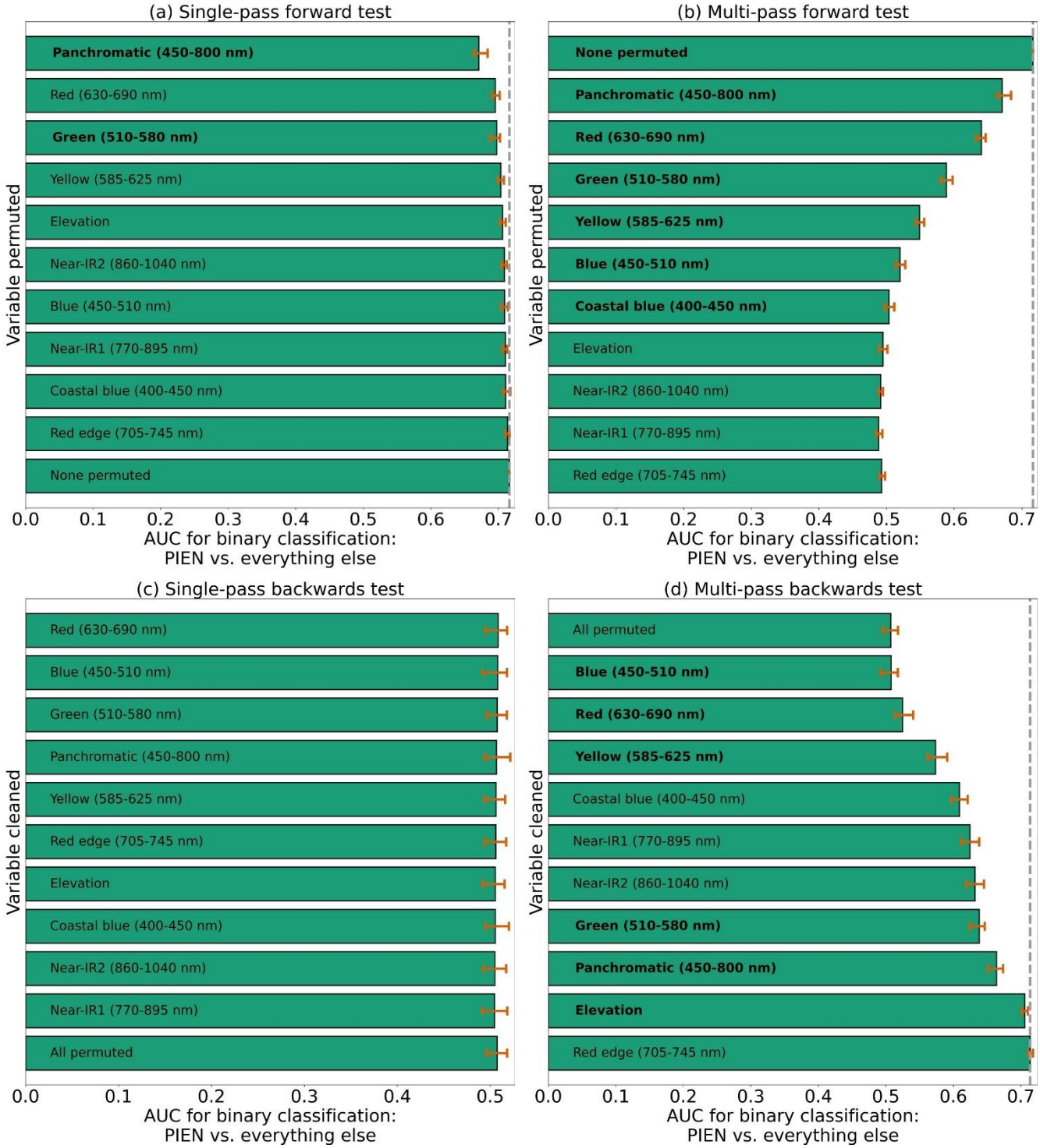


Figure S50. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying PIEN in the four-class model. See the caption for Figure S42 for more details.

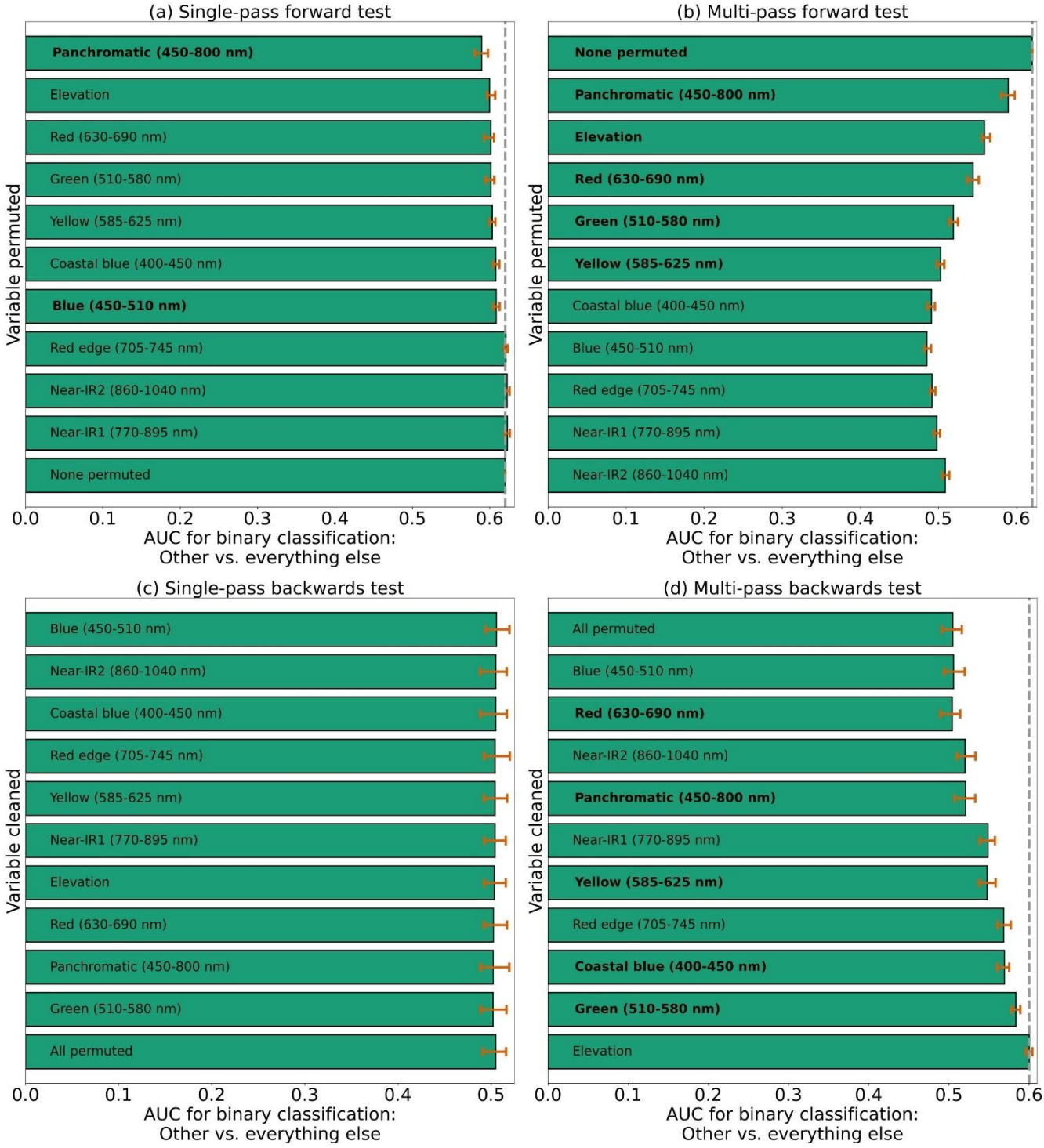


Figure S51. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying Other (non-conifer species) in the four-class model. See the caption for Figure S42 for more details.

S3.3 Two-Class Model Permutation Tests

Figure 13 of the main body shows results of the permutation test, for the two-class model, using the Gerrity score to consider discrimination among all species. Figure S52 is analogous but using the AUC metric.

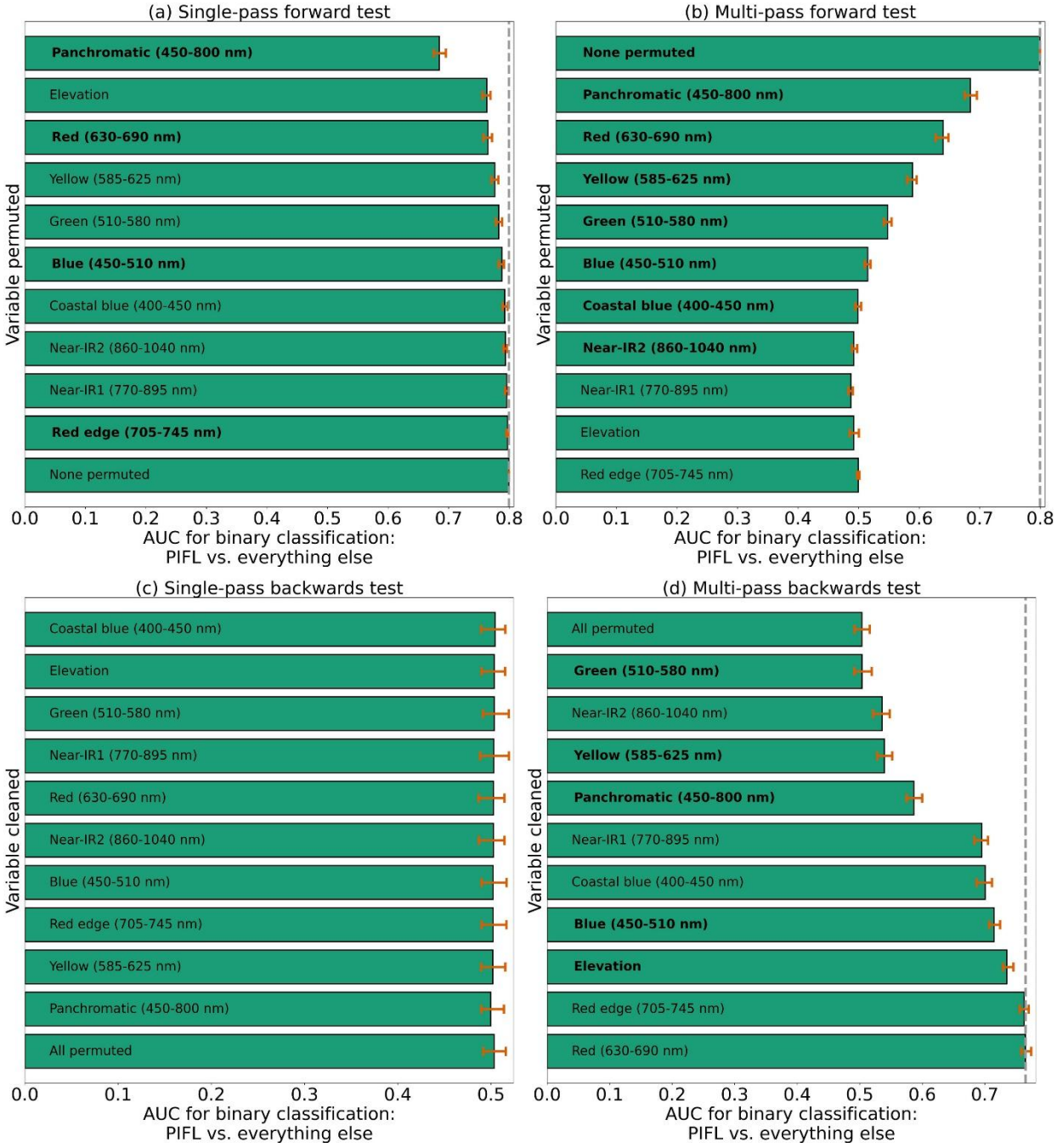


Figure S52. Results from each variety of the permutation test to assess predictor importance specific to correctly classifying limber pine (PIFL) in the two-class model. See the caption for Figure S42 for more details.

S5 Best Hits and Worst Confusion Between Two Species

The following sections provide examples of best hits and worst cases of confusion between two species. Examining cases of successes and failures is also a useful approach to understand what the CNN models are relying on for their predictions, as well as where they may be struggling. This process may also help with gauging how well the models are likely to generalize to other areas (in this case other treeline sites). Only examples from the six-class model are shown here, due to space constraints.

Figures S52-S62 in section S5.1 provide examples of best hits for each of the species in the six-class model. The figures are analogous to Figure 7 of the main manuscript. Figures S63-S75 in section S5.2 provide examples of cases of worst confusion between pairs of species in the six-class model that were commonly confused (based on the confusion matrix – Figure 6 in the main manuscript). These worst confusion figures are analogous to Figure 8 of the main manuscript.

S5.1 Best Hits

Figure S52 is an example of a best hit for limber pine (PIFL). The other example for PIFL is Figure 7 of the main manuscript. Both examples are cases where limber pine was the majority species in the area, and the community structure is characteristic of limber pine communities: relatively dispersed, smaller individuals (rather than large, contiguous krummholz mats).

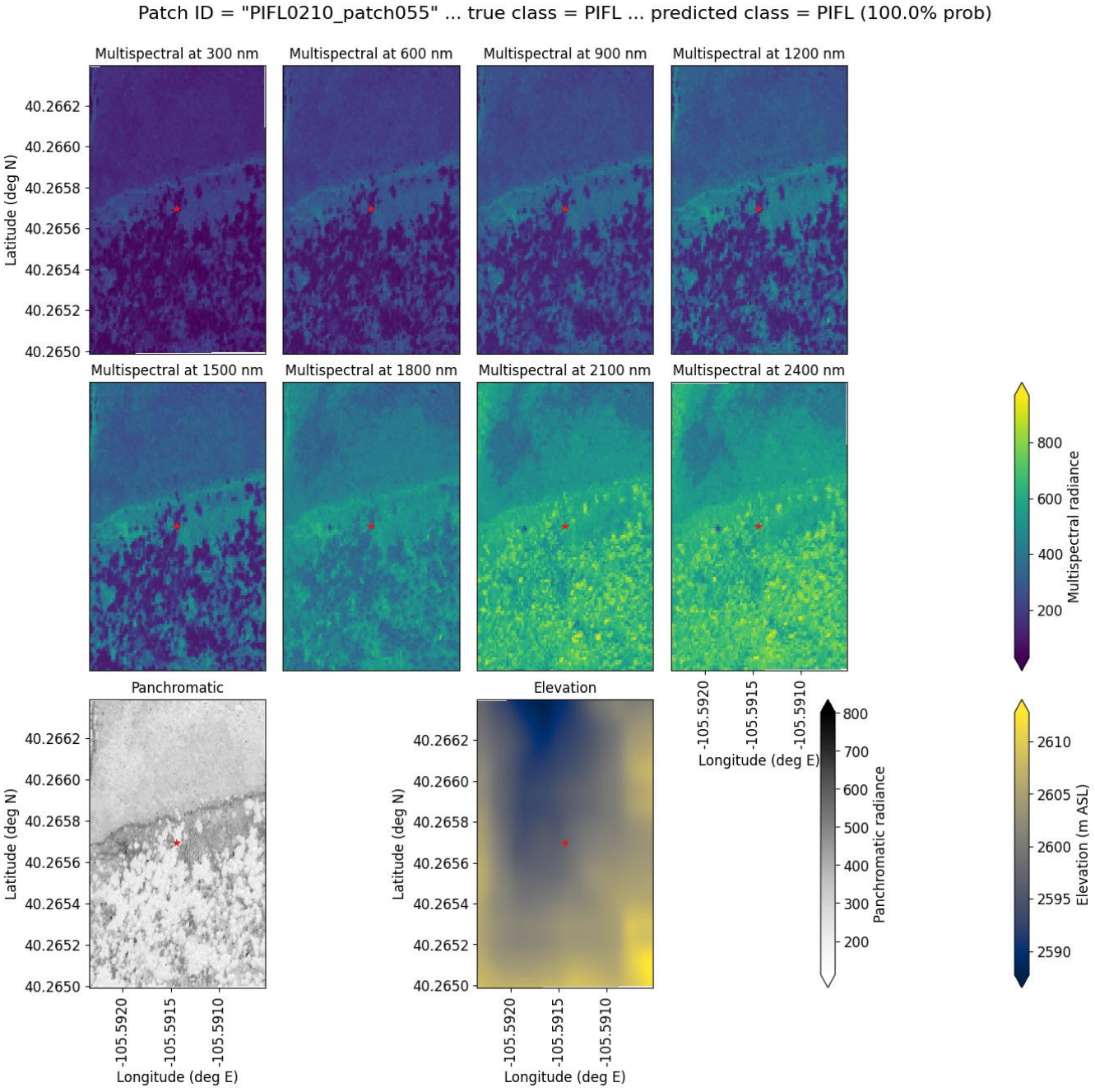


Figure S52. An example of a "best hit" classification of PIFL from the six-class model, where the model correctly predicted PIFL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. All eight multispectral bands, the panchromatic band, and the DEM are shown. The red star in the center of each image patch is the pixel being classified. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S53 and S54 are examples of best hits for Engelmann spruce (PIEN). In the first case, the pixel is in an island of spruce in an otherwise limber-pine-dominated area, which speaks to the model's skill. In the second case, the pixel was part of a larger krummholz patch amid other large krummholz patches in a higher area of treeline at the Longs Peak study site.

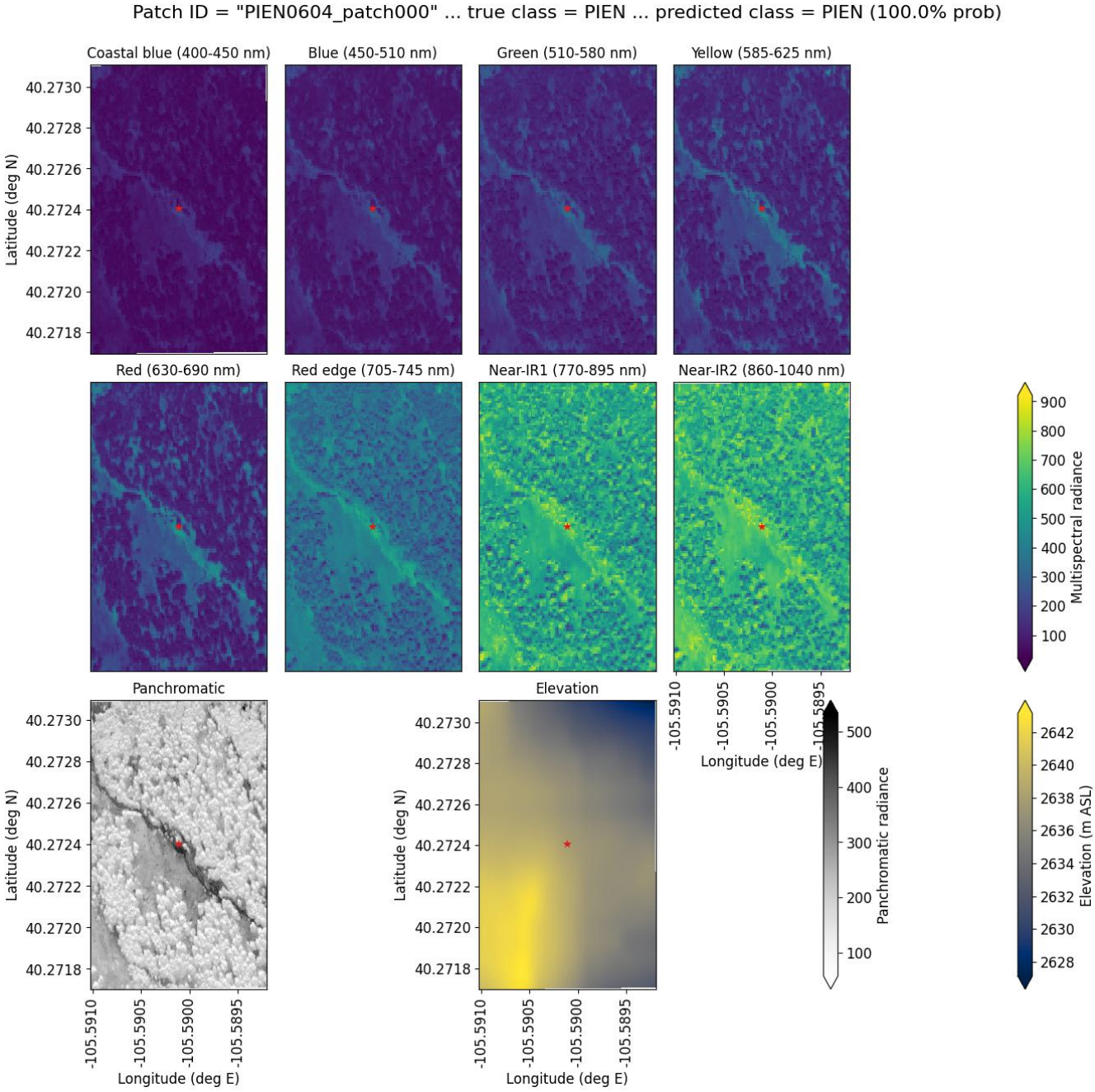


Figure S53. An example of a "best hit" classification of PIEN from the six-class model, where the model correctly predicted PIEN with 100% probability. This example (image chip or patch) is from the Battle Mountain study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

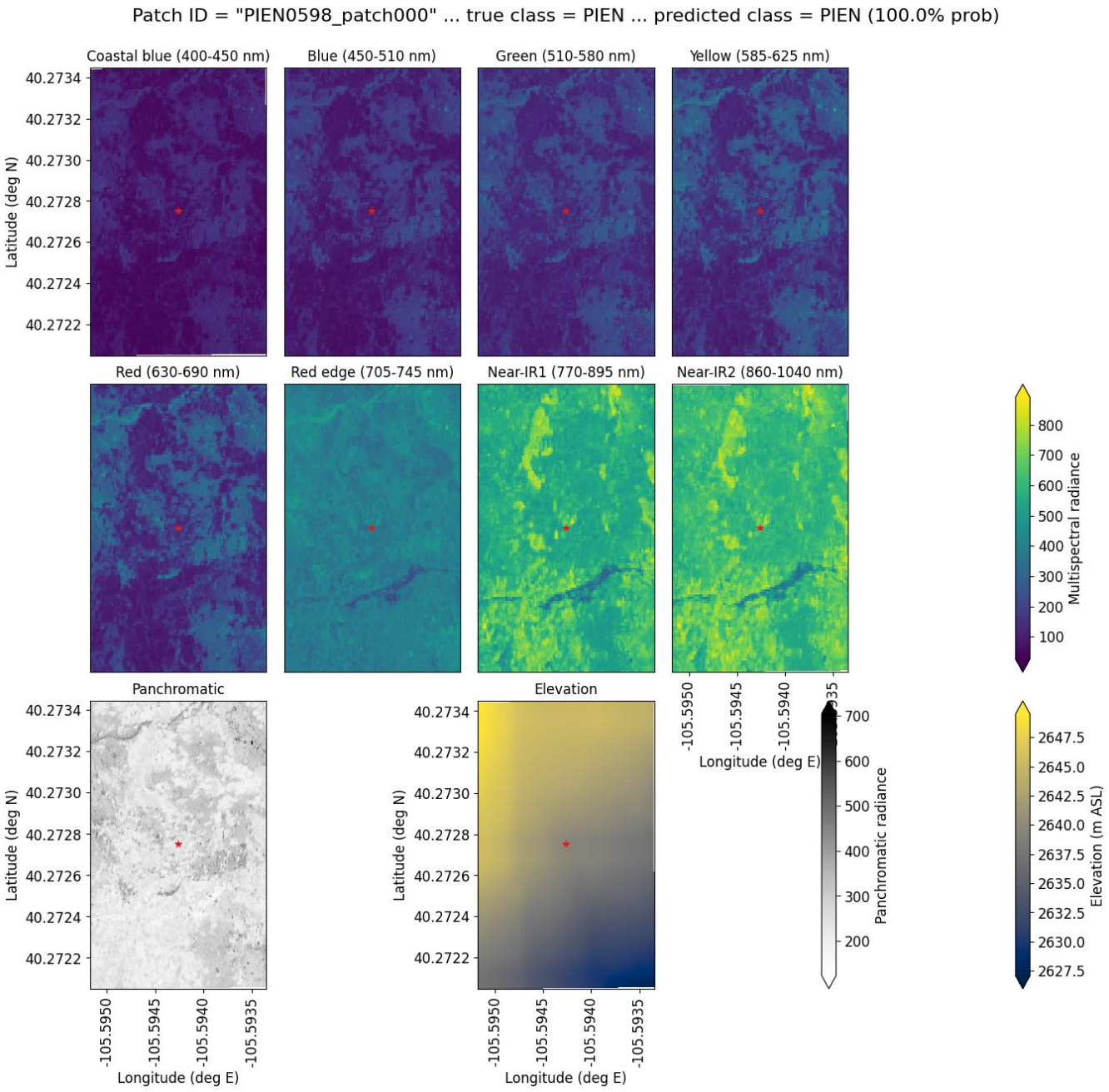


Figure S54. An example of a "best hit" classification of PIEN from the six-class model, where the model correctly predicted PIEN with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S55 and S56 are examples of best hits for subalpine fir (ABLA). In the first case, the pixel is part of a small tree island adjacent to the E Longs Peak trail. In the second case, the pixel is part of a very large, mixed tree island in the higher part of the treeline ecotone at the Longs Peak study site.

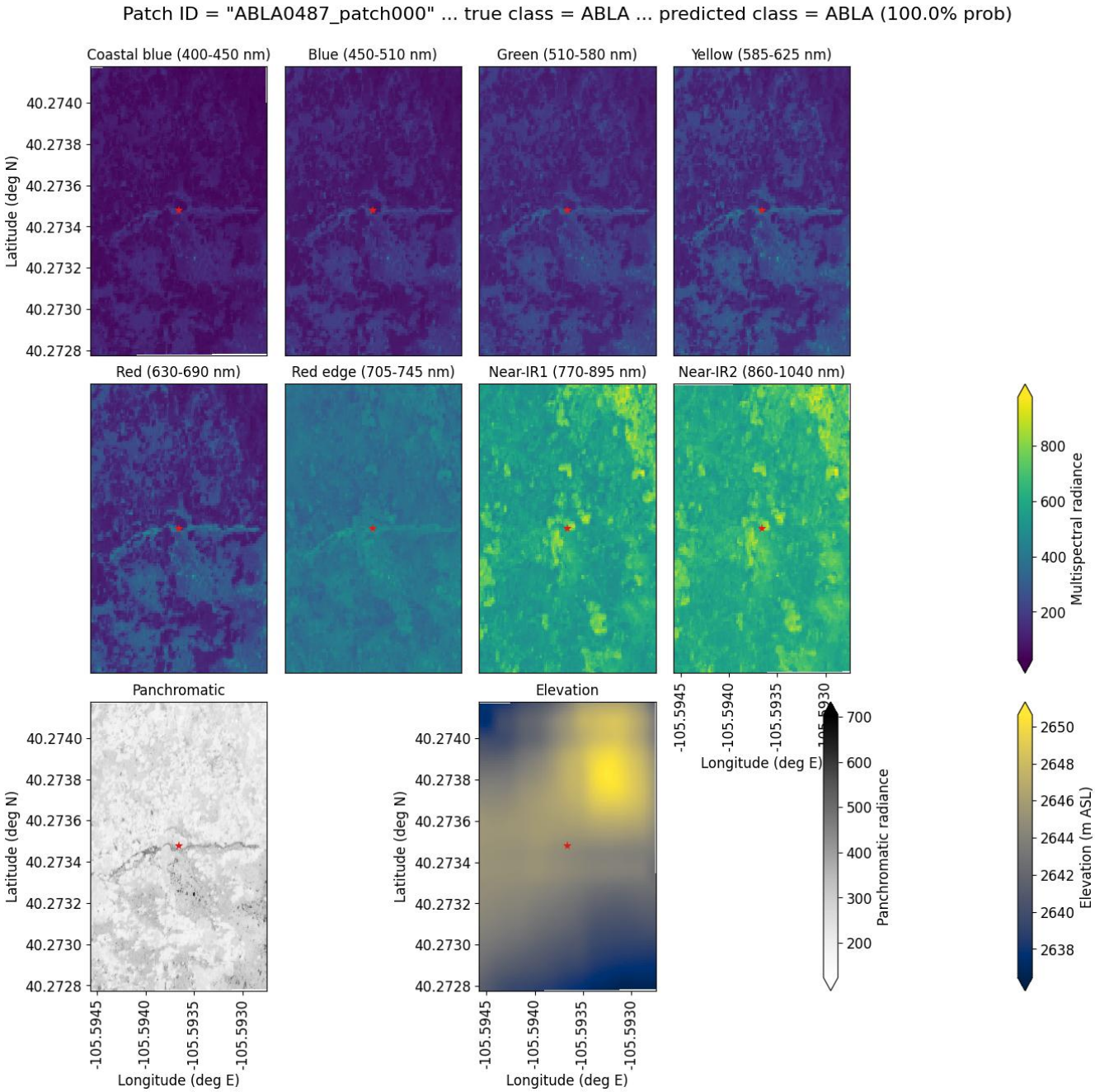


Figure S55. An example of a "best hit" classification of ABLA from the six-class model, where the model correctly predicted ABLA with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

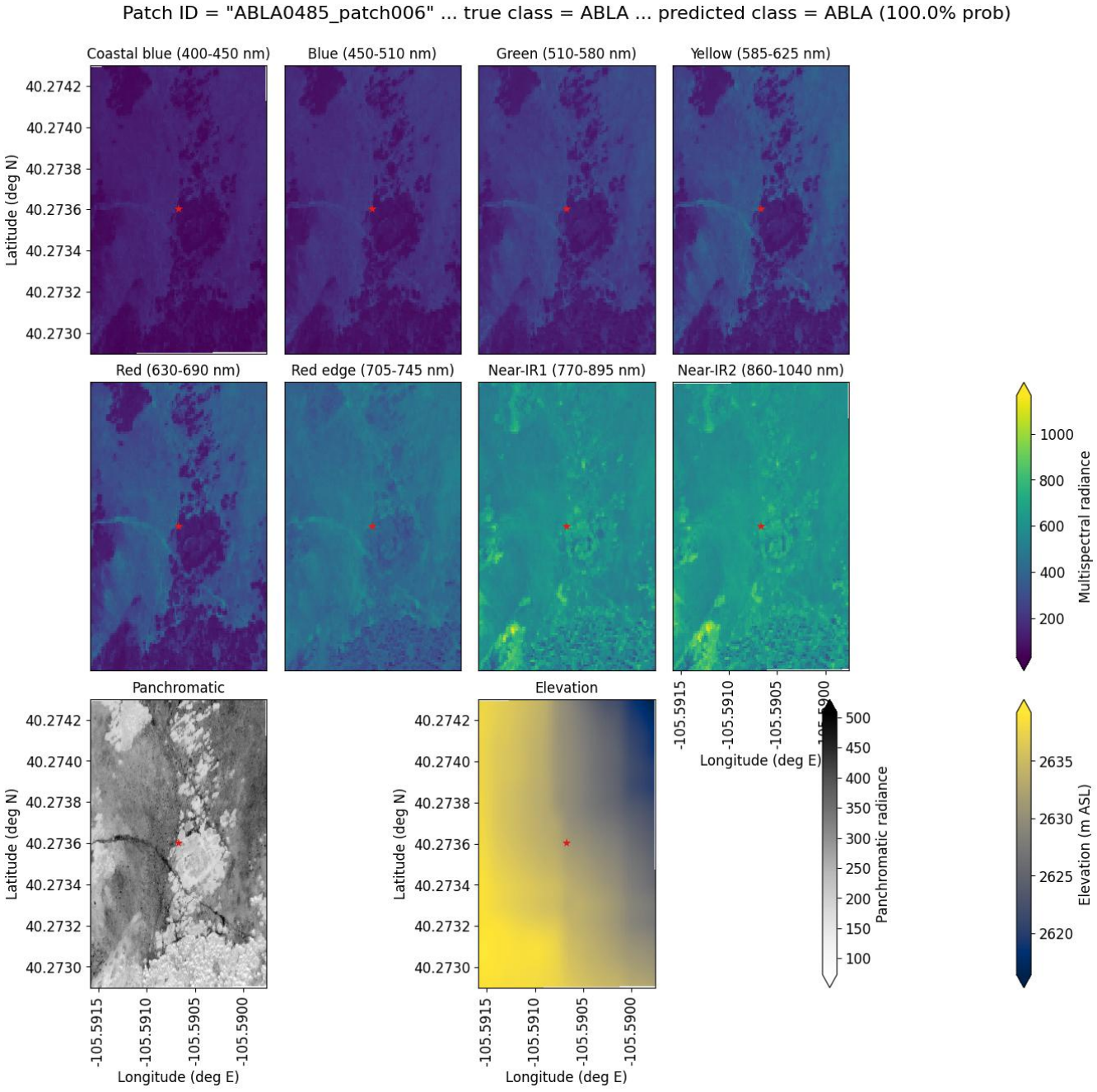


Figure S56. An example of a "best hit" classification of ABLA from the six-class model, where the model correctly predicted ABLA with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S57 and S58 are examples of best hits for glandular birch (BEGL). In the first case, the pixel is part of a very mixed community that included limber pine, Engelmann spruce, and subalpine fir. In the second case, the pixel is part of elongated islands of glandular birch above a large patch of mixed krummholz species.

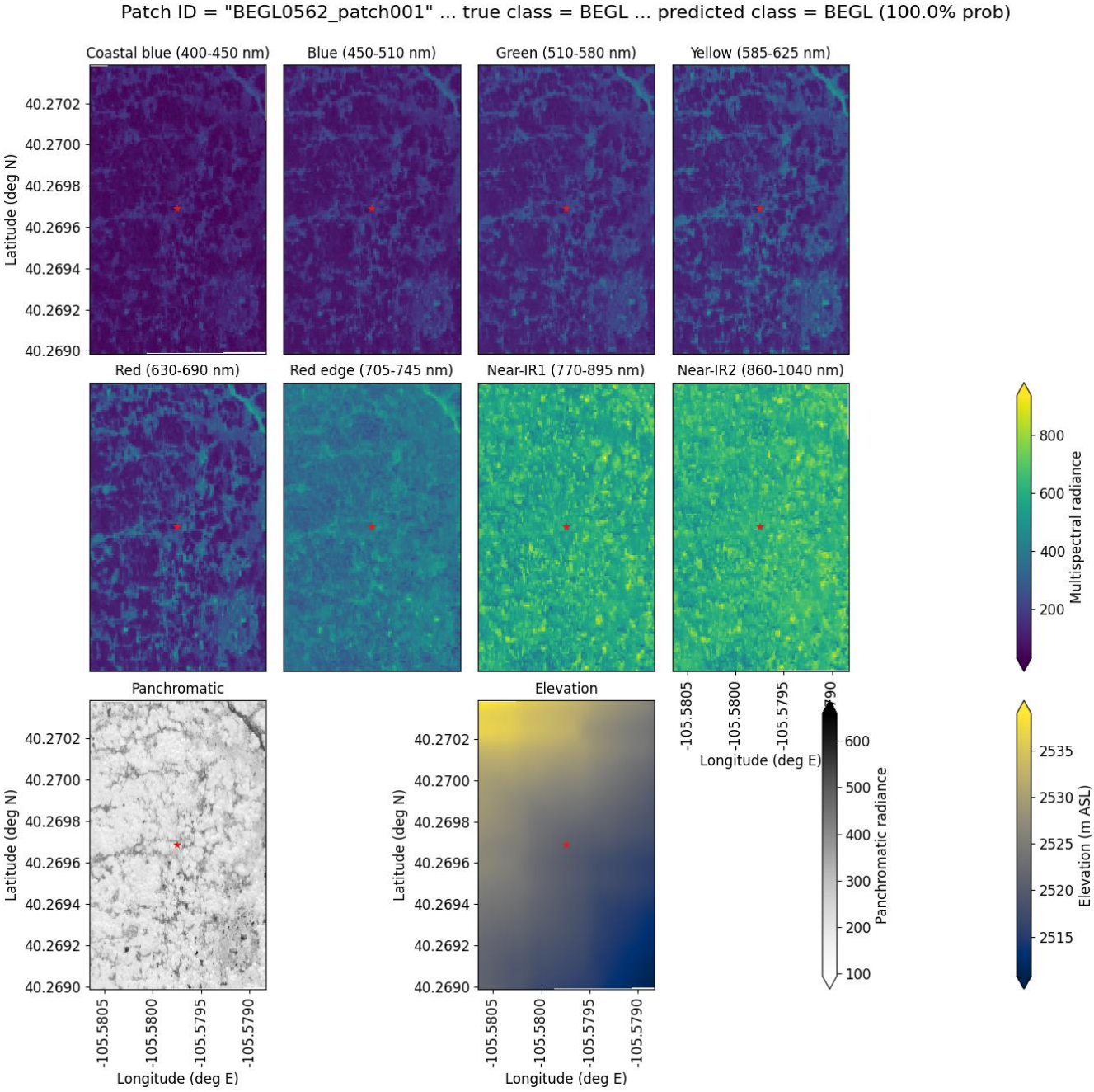


Figure S57. An example of a "best hit" classification of BEGL from the six-class model, where the model correctly predicted BEGL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} µm^{-1}.

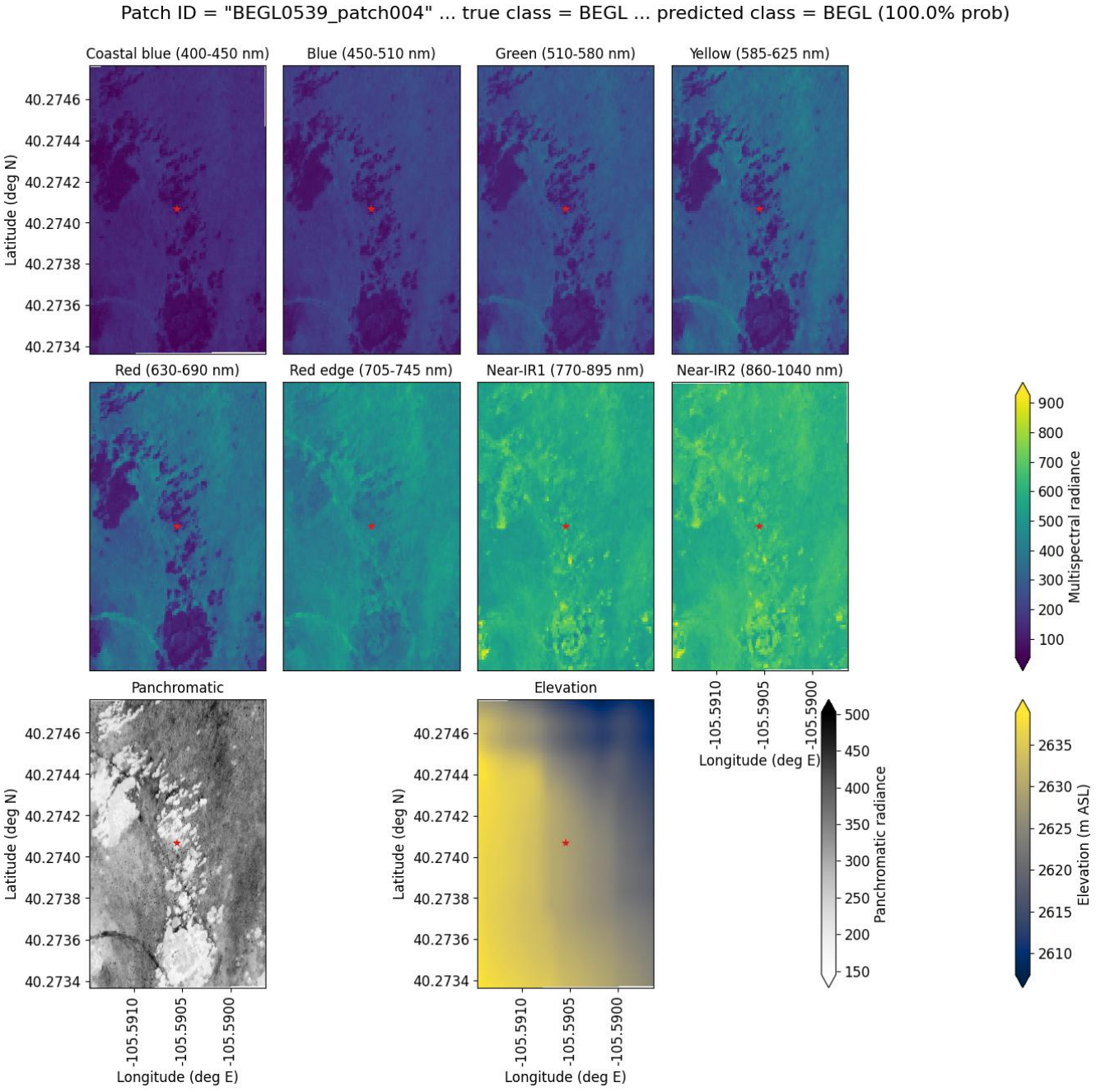


Figure S58. An example of a "best hit" classification of BEGL from the six-class model, where the model correctly predicted BEGL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} µm^{-1}.

Figures S59 and S60 are examples of best hits for quaking aspen (POTR). In both cases, the pixel was a part of dense patches of aspen amid a complex community including other species.

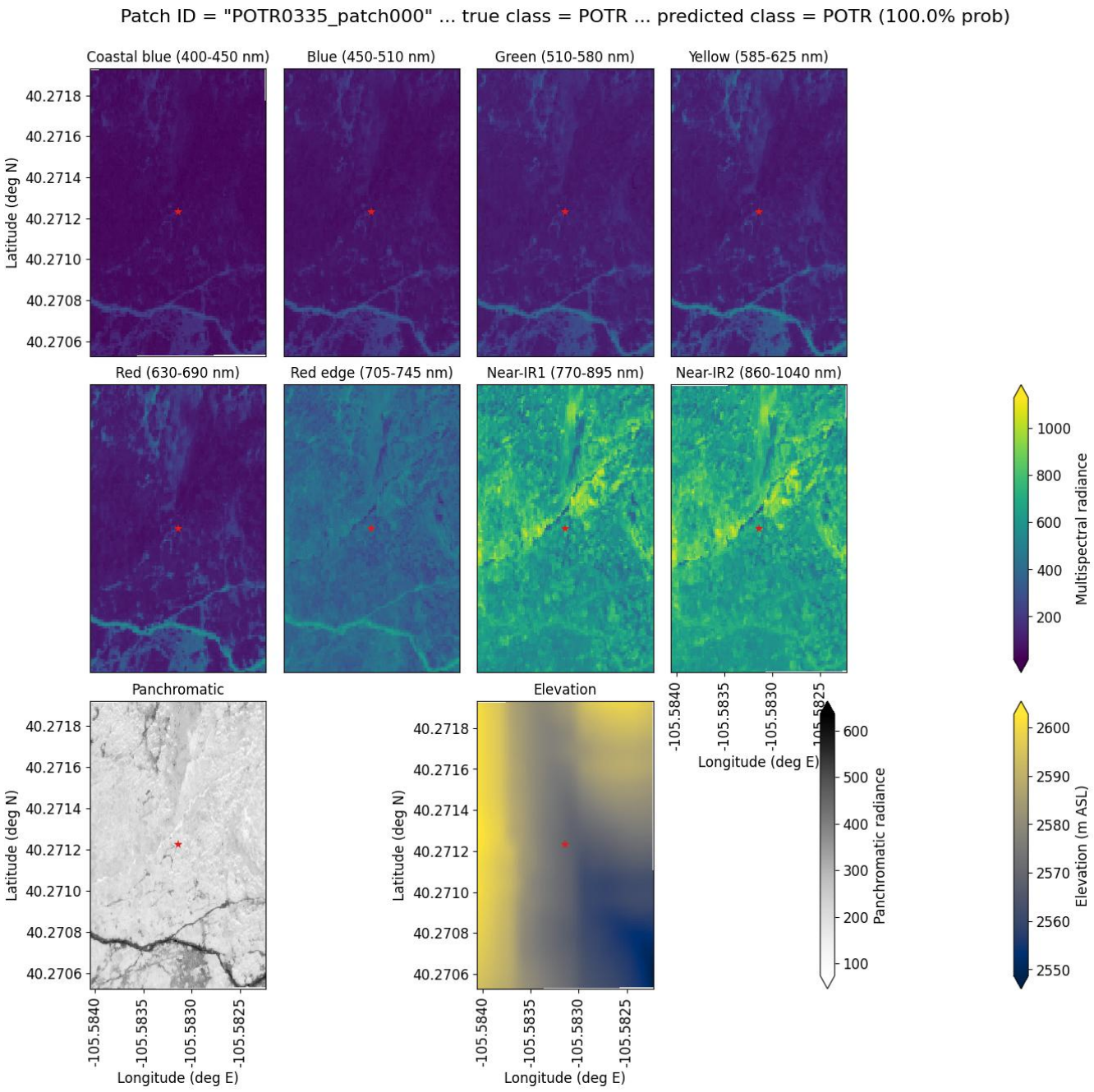


Figure S59. An example of a "best hit" classification of POTR from the six-class model, where the model correctly predicted POTR with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

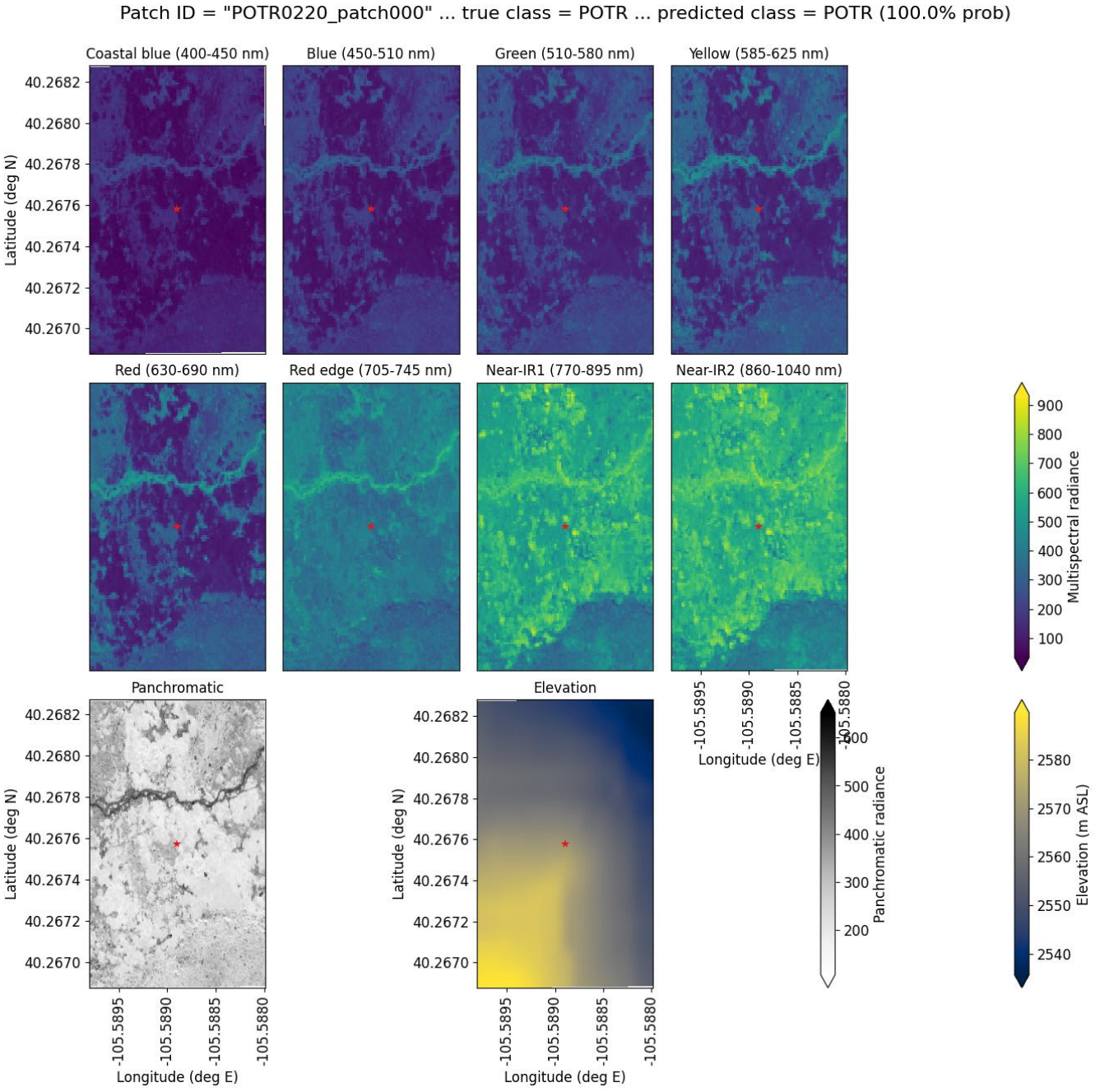


Figure S60. An example of a "best hit" classification of POTR from the six-class model, where the model correctly predicted POTR with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S61 and S62 are examples of best hits for willow (*Salix spp.*). In both cases, the pixel was part of a mixed community of species near a trail.

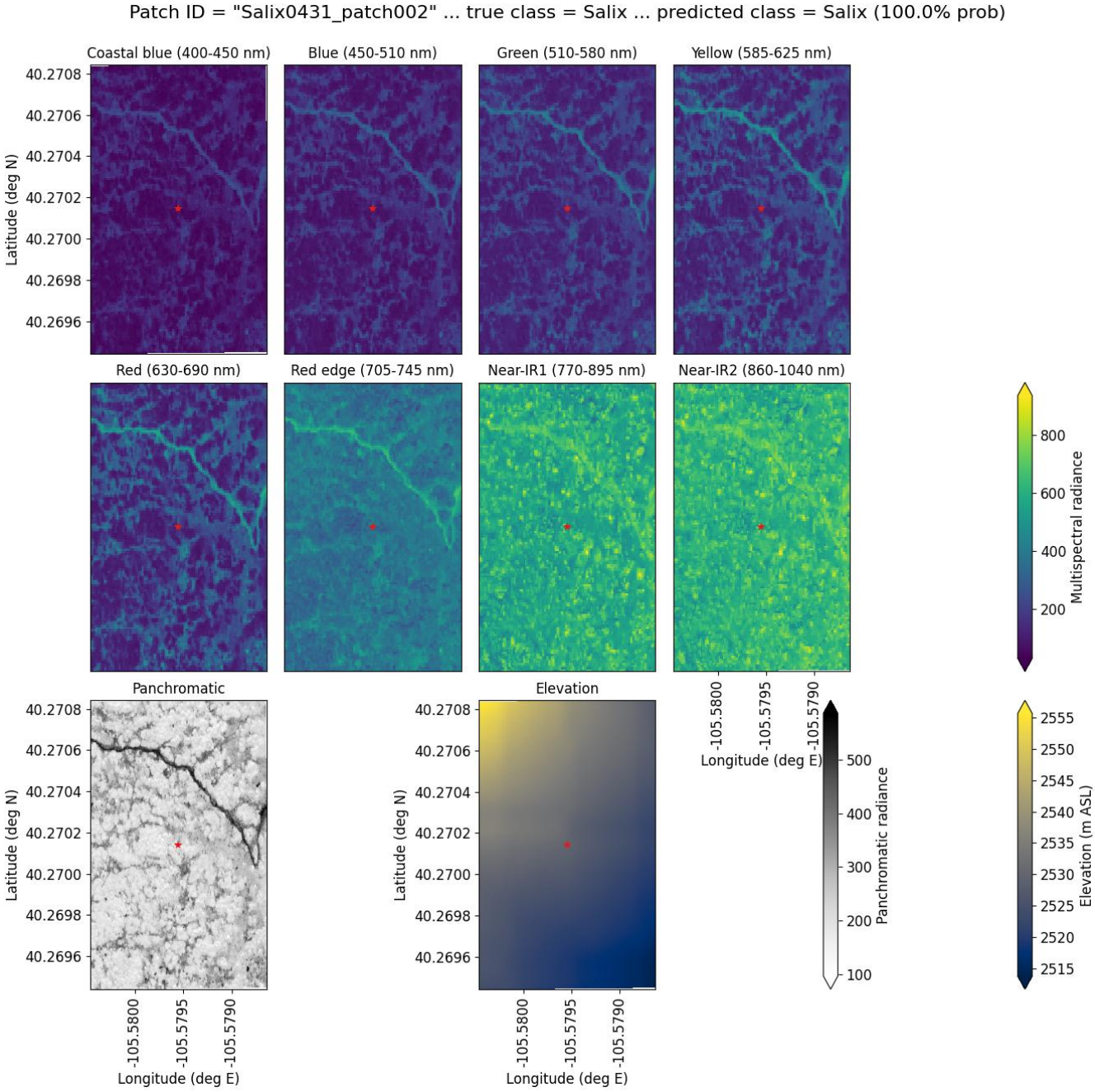


Figure S61. An example of a "best hit" classification of *Salix* from the six-class model, where the model correctly predicted *Salix* with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

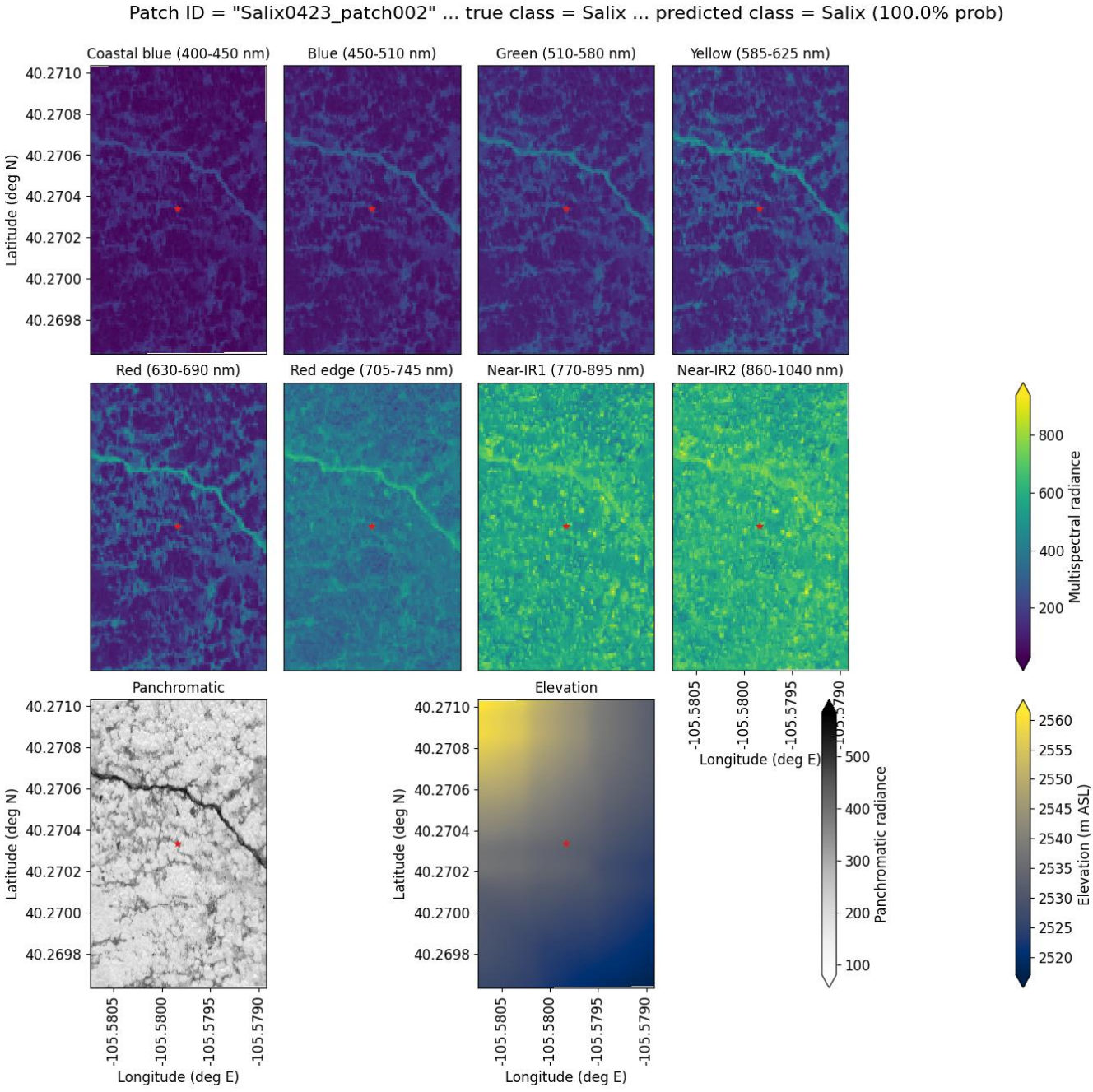


Figure S62. An example of a "best hit" classification of *Salix* from the six-class model, where the model correctly predicted *Salix* with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

In all of the best hits cases explored here, the pixels were part of typical community types/situations for each species. However, the best hits also included examples of the model succeeding in complex communities—small and large tree islands, and cases with denser communities with more mixed species composition.

S5.2 Worst Confusion

Figure S63 is an example of worst confusion where limber pine (PIFL) was the observed species and willow (*Salix*) was the predicted species. We can see that in both cases, the pixel was in community types that are emblematic of limber pine communities—texturally speaking—but that also occasionally contain willow. In these cases, the elevation data would not be helpful because while willow is more abundant near streams, it is not exclusively found near streams. The CNN would need to make use of the multispectral data, but these species are only distinguishable in the N-IR1 and N-IR2 bands (Table S1).

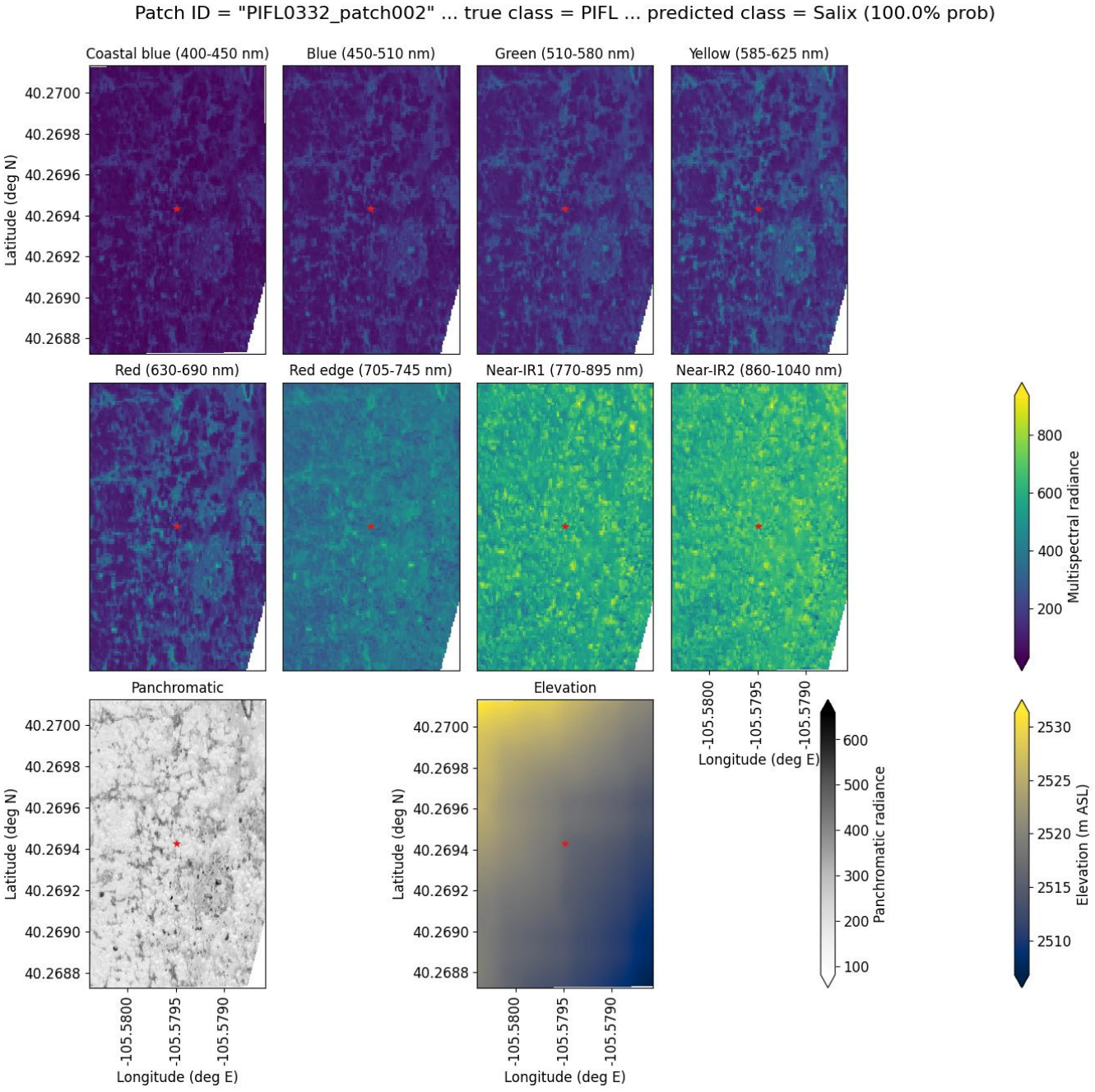


Figure S63. An example of a "worst confusion" classification of PIFL as *Salix* from the six-class model, where the model incorrectly predicted PIFL as *Salix* with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S64 and S65 are examples of worst confusion between PIFL and Engelmann spruce (PIEN). Figure S64 shows a case where PIFL was the observed species and PIEN was predicted. The community is dominated by PIFL and emblematic of that community type, but there were occasional PIENs in that community.

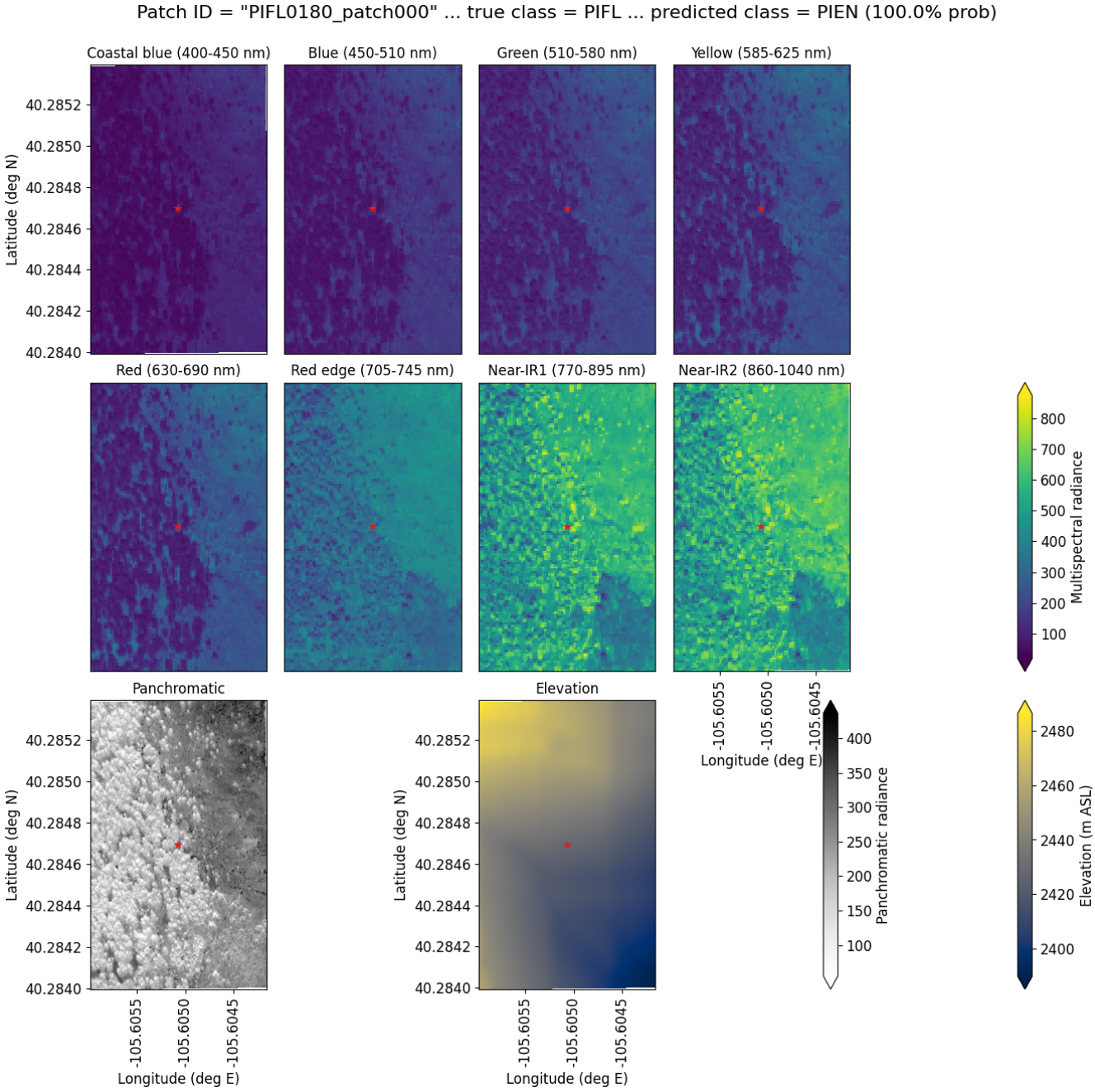


Figure S64. An example of a "worst confusion" classification of PIFL as PIEN from the six-class model, where the model incorrectly predicted PIFL as PIEN with 100% probability. This example (image chip or patch) is from the Battle Mountain study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S65 shows a case where PIEN was the observed species and PIFL was predicted. The community included larger krummholz mats and strings of glandular birch. PIFL was not common in this region of the study area, but the pixel in question is part of a smaller mat of krummholz PIEN.

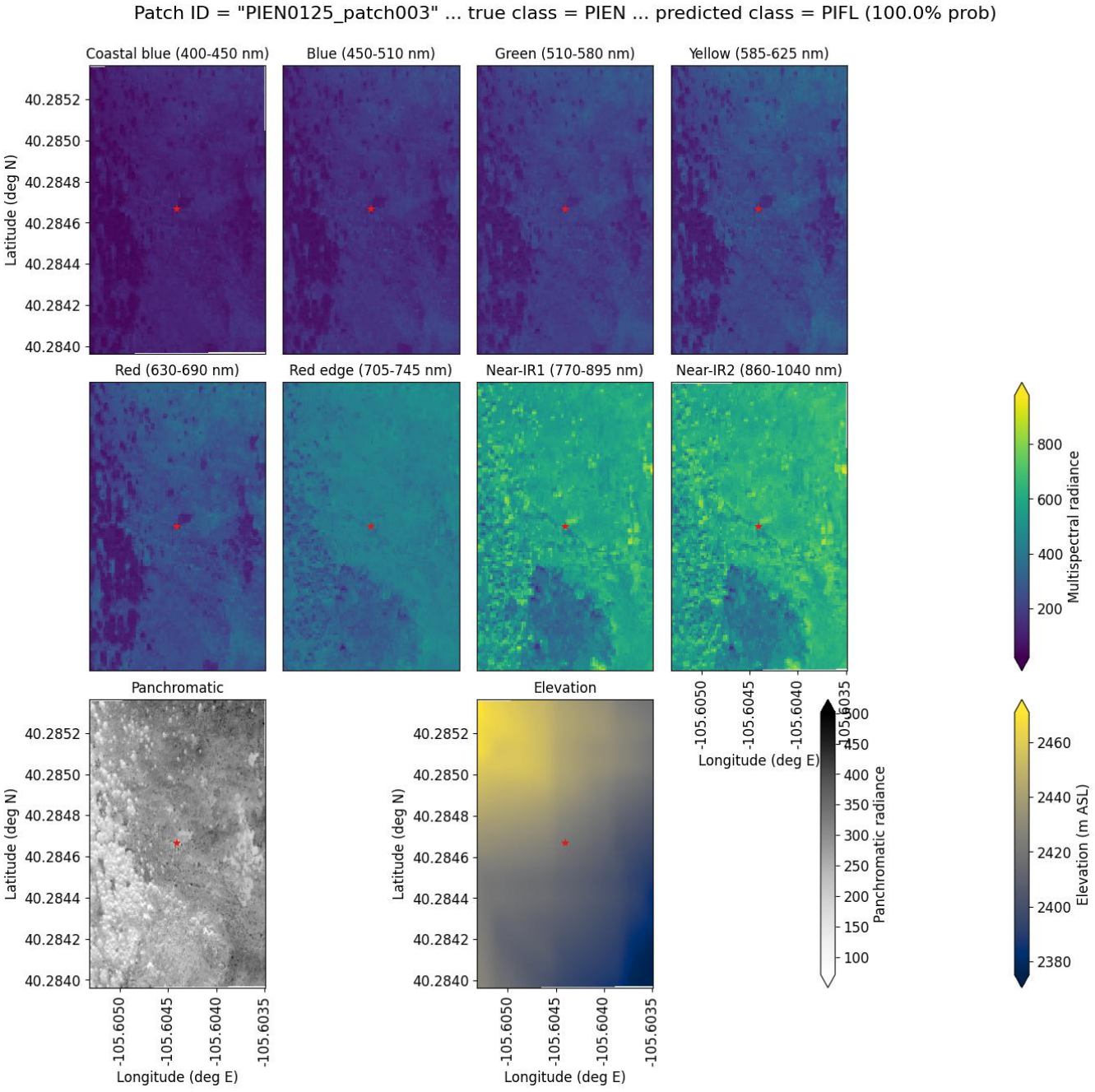


Figure S65. An example of a "worst confusion" classification of PIEN as PIFL from the six-class model, where the model incorrectly predicted PIEN as PIFL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S66 is an example of worst confusion where glandular birch (BEGL) was the observed species and PIFL was the predicted species. The pixel is part of larger patches of BEGL but is on the edge of the patch.

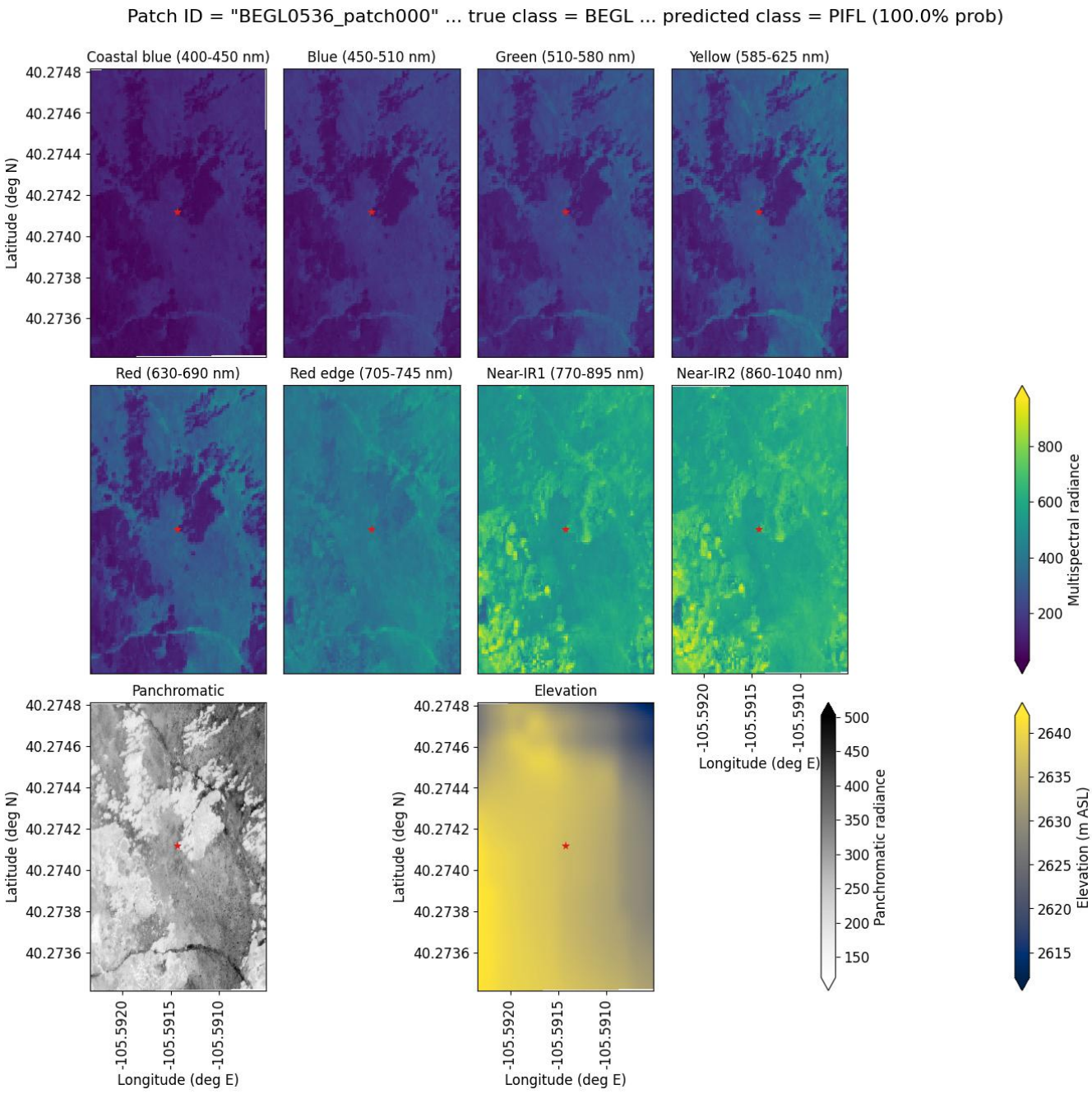
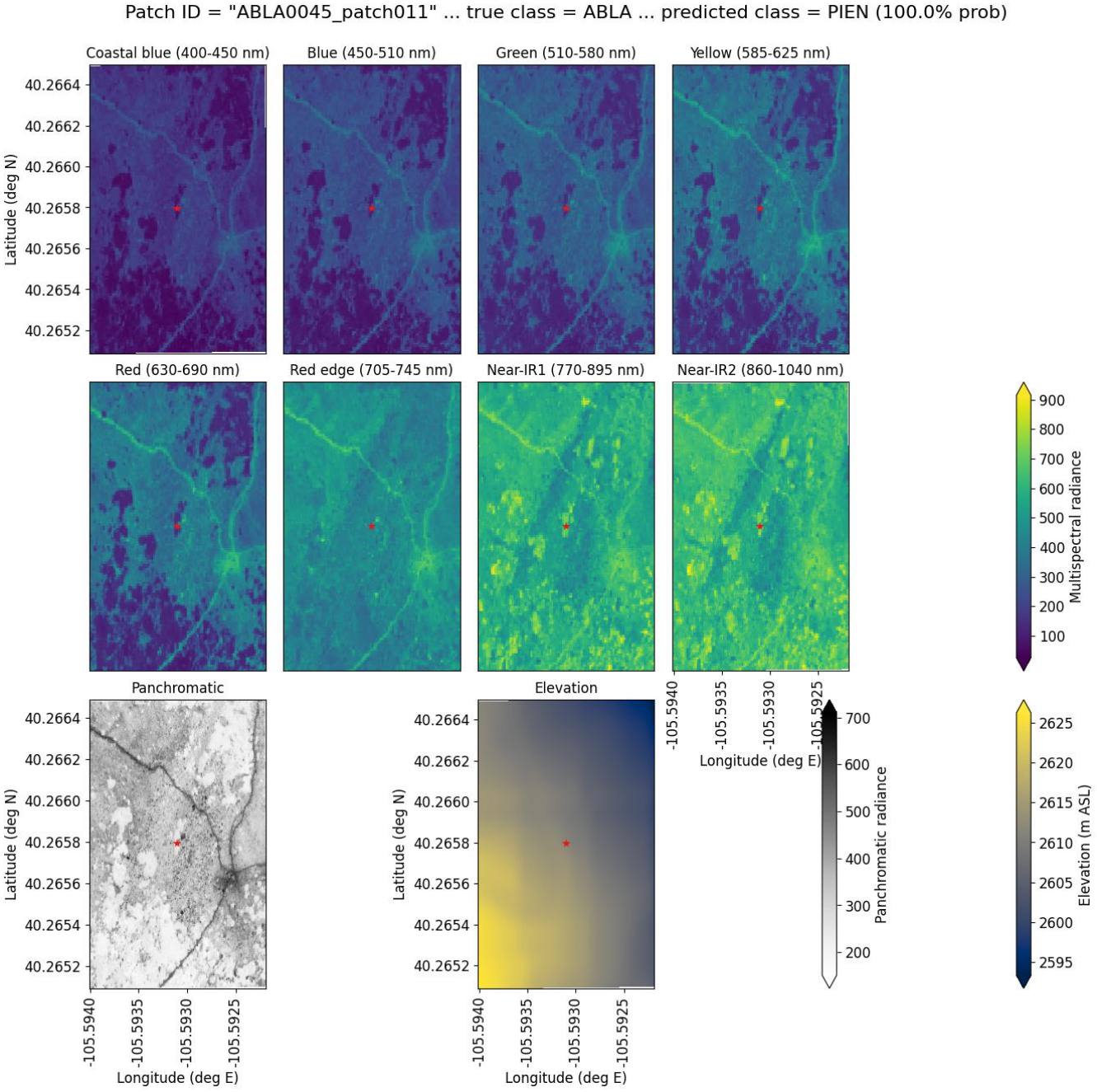


Figure S66. An example of a "worst confusion" classification of BEGL as PIFL from the six-class model, where the model incorrectly predicted BEGL as PIFL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

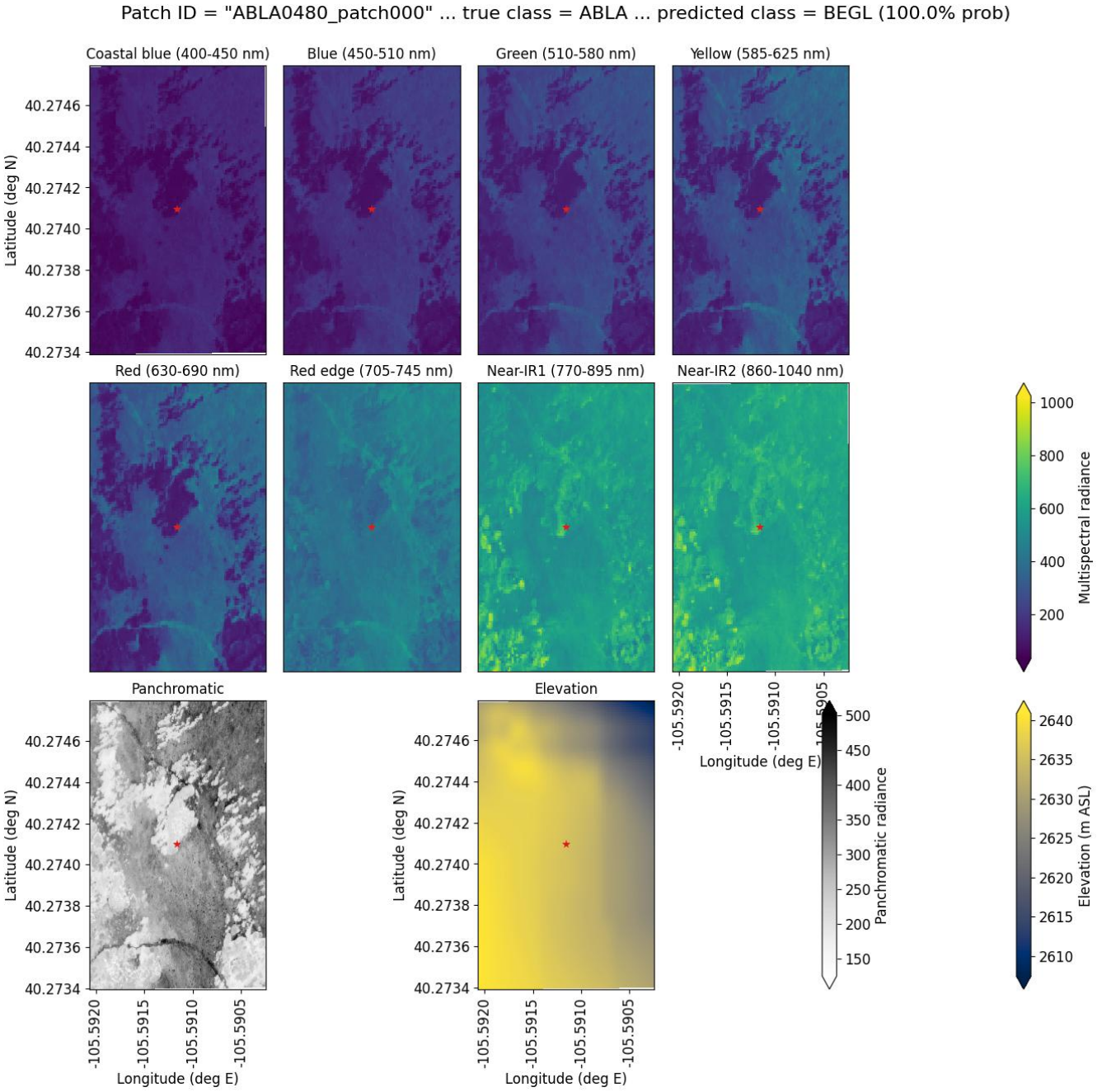
Figure S67 is an example of worst confusion where subalpine fir (ABLA) was the observed species and PIEN was the predicted species. Both ABLA and PIEN are often found in krummholz mats, sometimes very large and often of medium size as pictured in Figure S67. They are not easily distinguished in the panchromatic imagery, suggesting the model is not making use of the multispectral imagery to distinguish these species (also see permutation test Figures S43 and S45).



585

Figure S67. An example of a "worst confusion" classification of ABLA as PIEN from the six-class model, where the model incorrectly predicted ABLA as PIEN with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S68 is an example of worst confusion where ABLA was the observed species and BEGL was the predicted species. BEGL does grow in large, elongated clumps on the landscape. ABLA also grows in larger krummholz mats from time to time. If the model is relying too much on the panchromatic and elevation data, these species would become difficult to distinguish.



595 **Figure S68. An example of a "worst confusion" classification of ABLA as BEGL from the six-class model, where the model incorrectly predicted ABLA as BEGL with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.**

Figure S69 is an example of worst confusion where ABLA was the observed species and aspen (POTR) was the predicted species. The case is difficult to interpret and may simply be attributable to the small sample size for POTR. All of the POTR

samples were parts of larger patches of POTR, but the pixel in the example below is part of a smaller island more typical of ABLA.

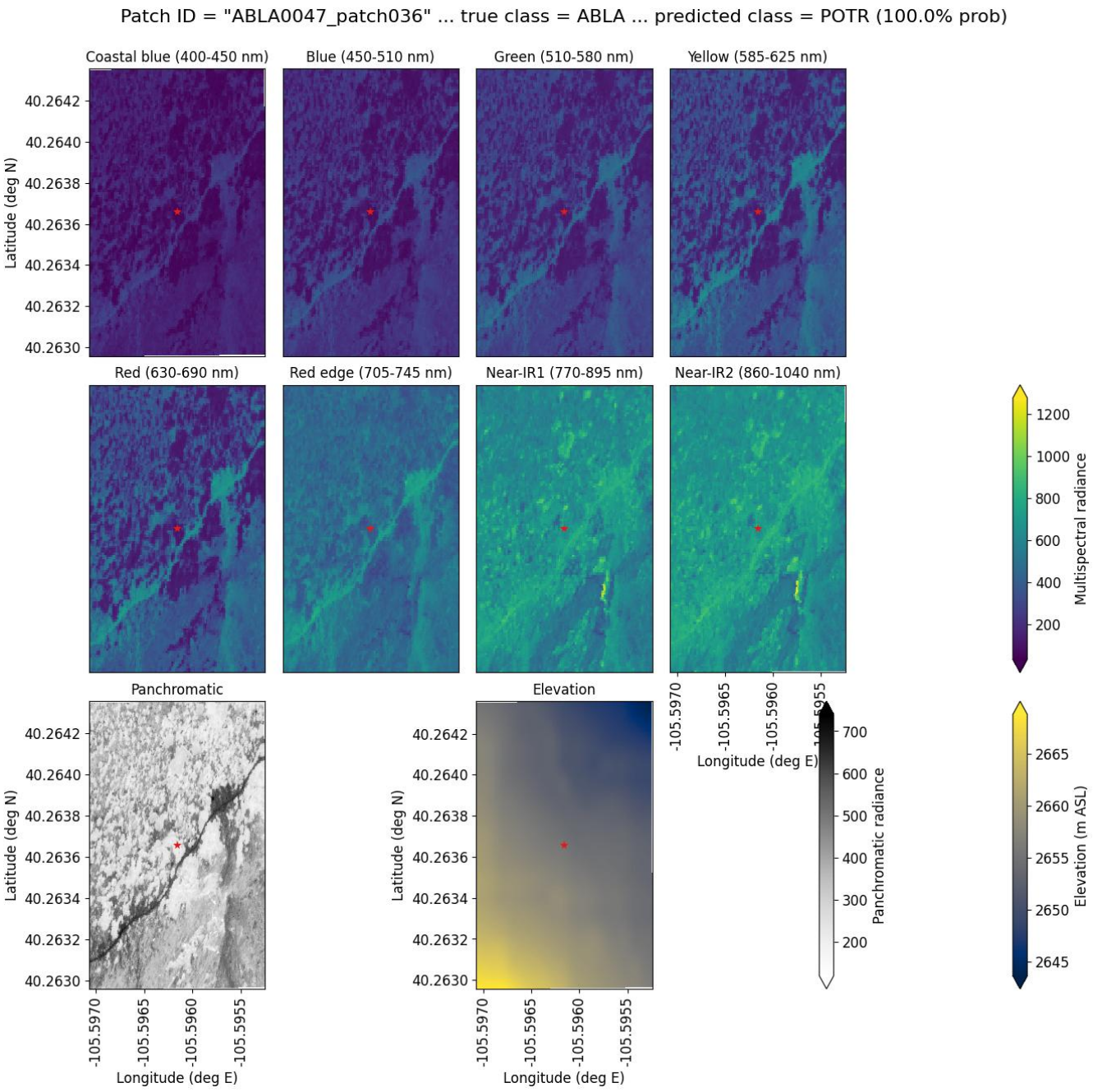


Figure S69. An example of a "worst confusion" classification of ABLA as POTR from the six-class model, where the model incorrectly predicted ABLA as POTR with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figures S70 and S71 are examples of worst confusion between PIEN and BEGL. Figure S70 shows a case where PIEN was the observed species and BEGL was predicted. In this case, the pixel is part of a krummholz PIEN at the Battle Mountain site, where PIEN was a minor component. The model may have learned that BEGL is more common at that site, which is higher elevation than the Longs Peak site.

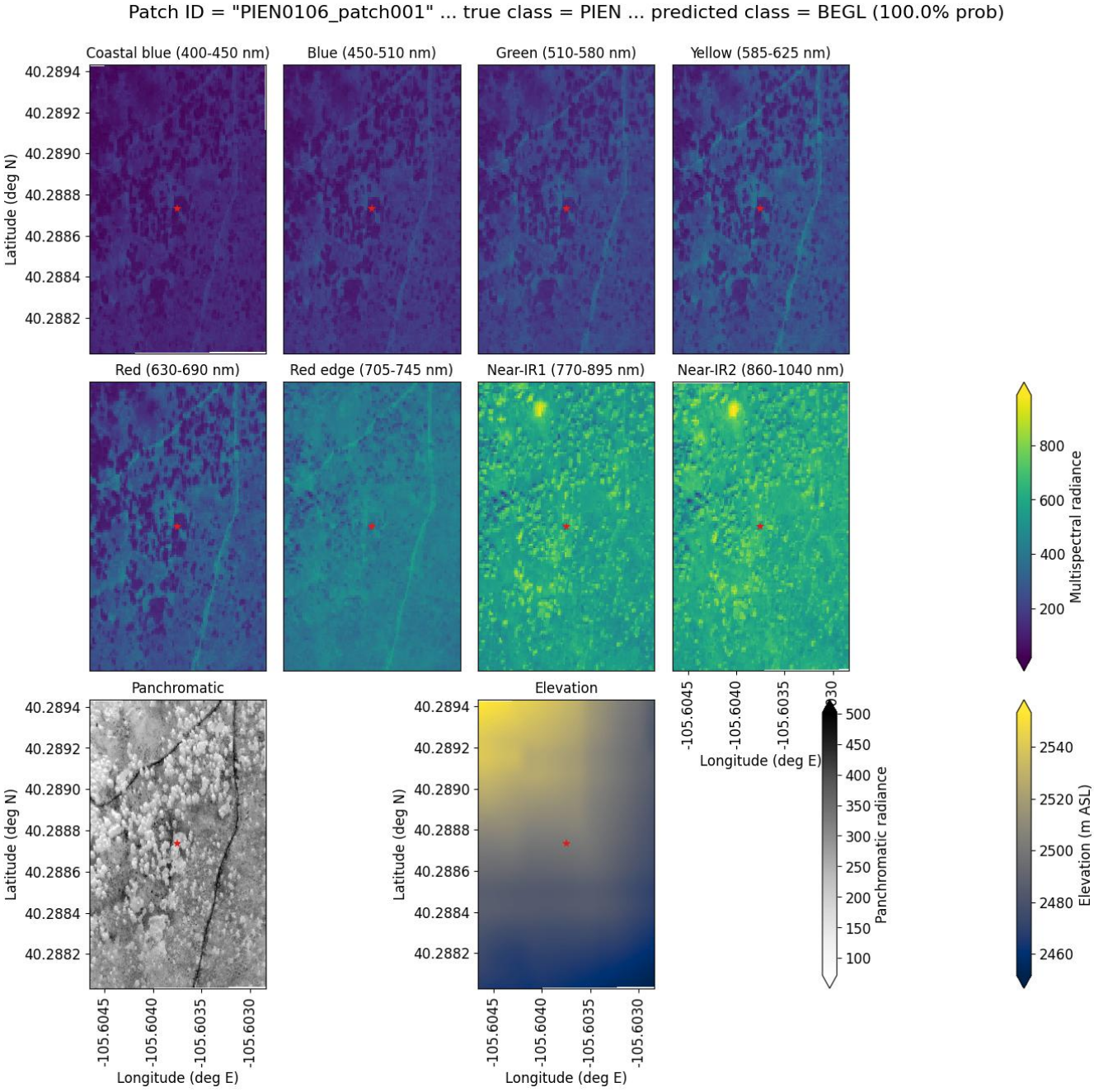


Figure S70. An example of a "worst confusion" classification of PIEN as BEGL from the six-class model, where the model incorrectly predicted PIEN as BEGL with 100% probability. This example (image chip or patch) is from the Battle Mountain study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S71 shows a case where BEGL was the observed species and PIEN was predicted. The case is an understandable moment of confusion. A smaller patch of BEGL at the longs peak site amid other small krummholz mats of PIEN and ABLA might be expected to contain PIEN or ABLA. The textural and elevational information would lead the model astray here.

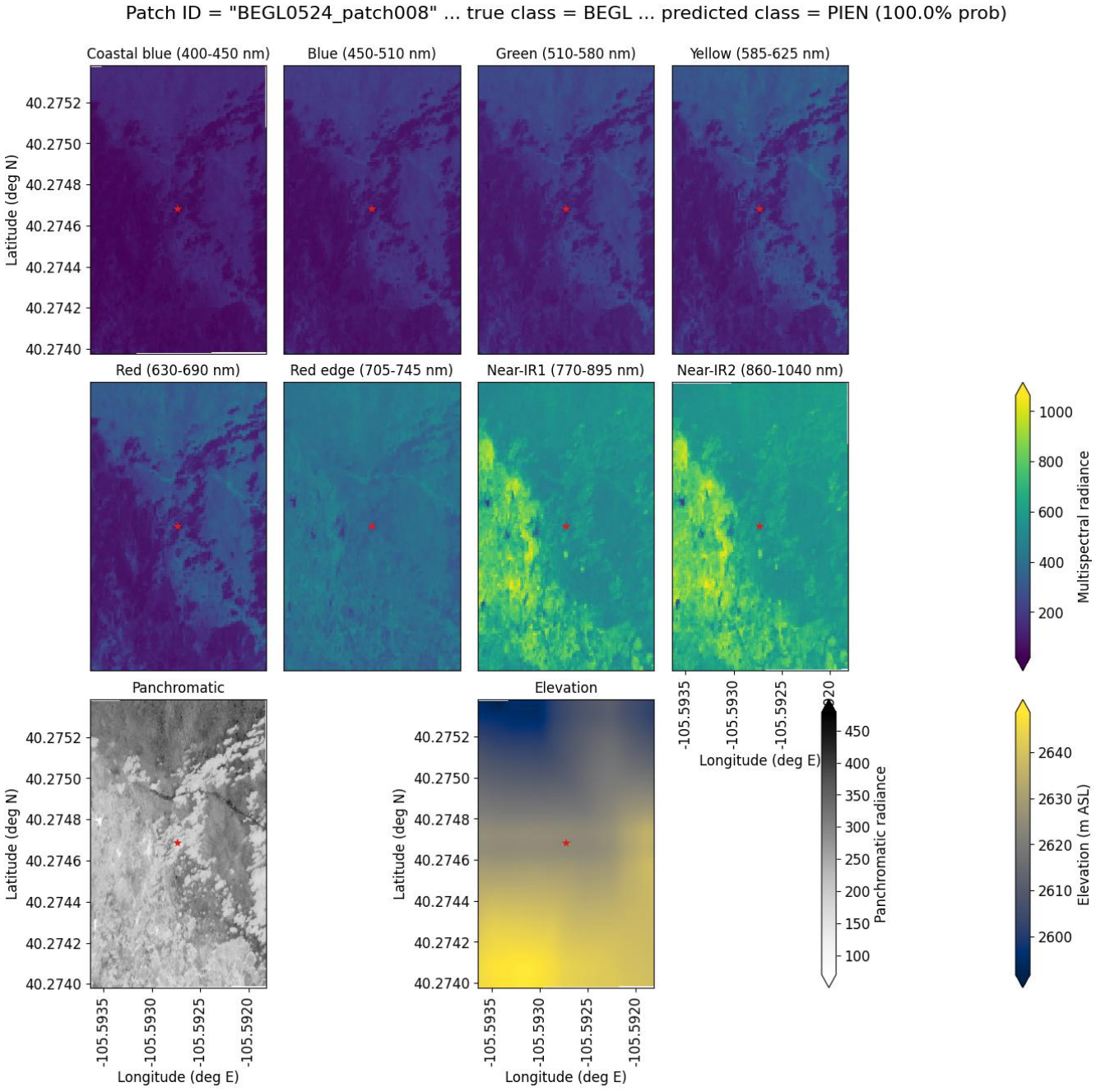
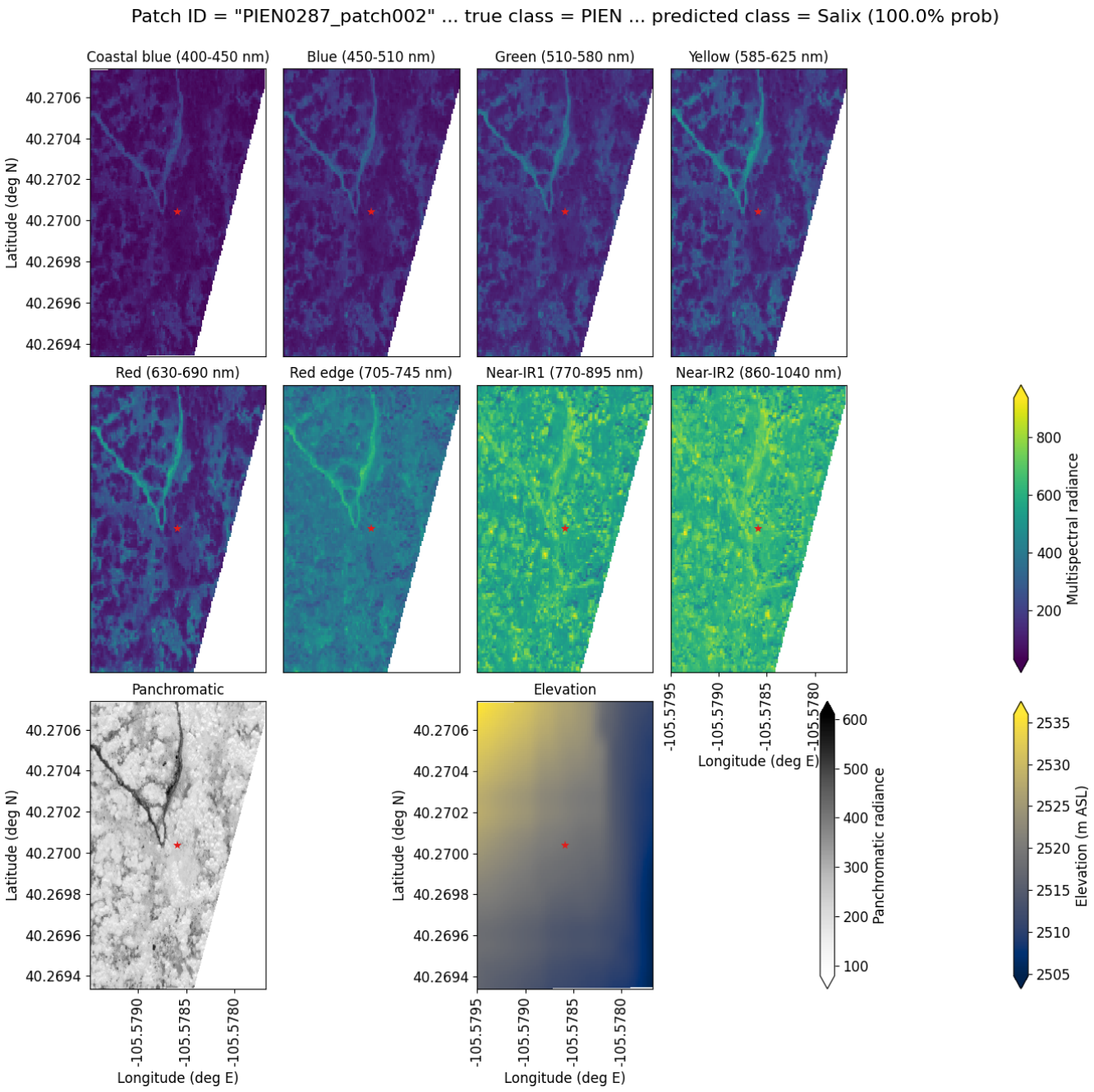


Figure S71. An example of a "worst confusion" classification of BEGL as PIEN from the six-class model, where the model incorrectly predicted BEGL as PIEN with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

620 Figures S72 and S73 are examples of worst confusion between PIEN and *Salix*. Figure S72 shows a case where PIEN was the observed species and *Salix* was predicted. In this case, the pixel is close to other locations where willow was found in

abundance at the Longs Peak site (similar elevation), and the community structure is similar. However, the community contains mostly Engelmann spruce and subalpine fir in this area.



625 **Figure S72. An example of a "worst confusion" classification of PIEN as *Salix* from the six-class model, where the model incorrectly predicted PIEN as *Salix* with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.**

Figure S73 shows a case where *Salix* was the observed species and PIEN was predicted. This is a case where an odd willow was found amid an area with mostly Engelmann spruce and subalpine fir.

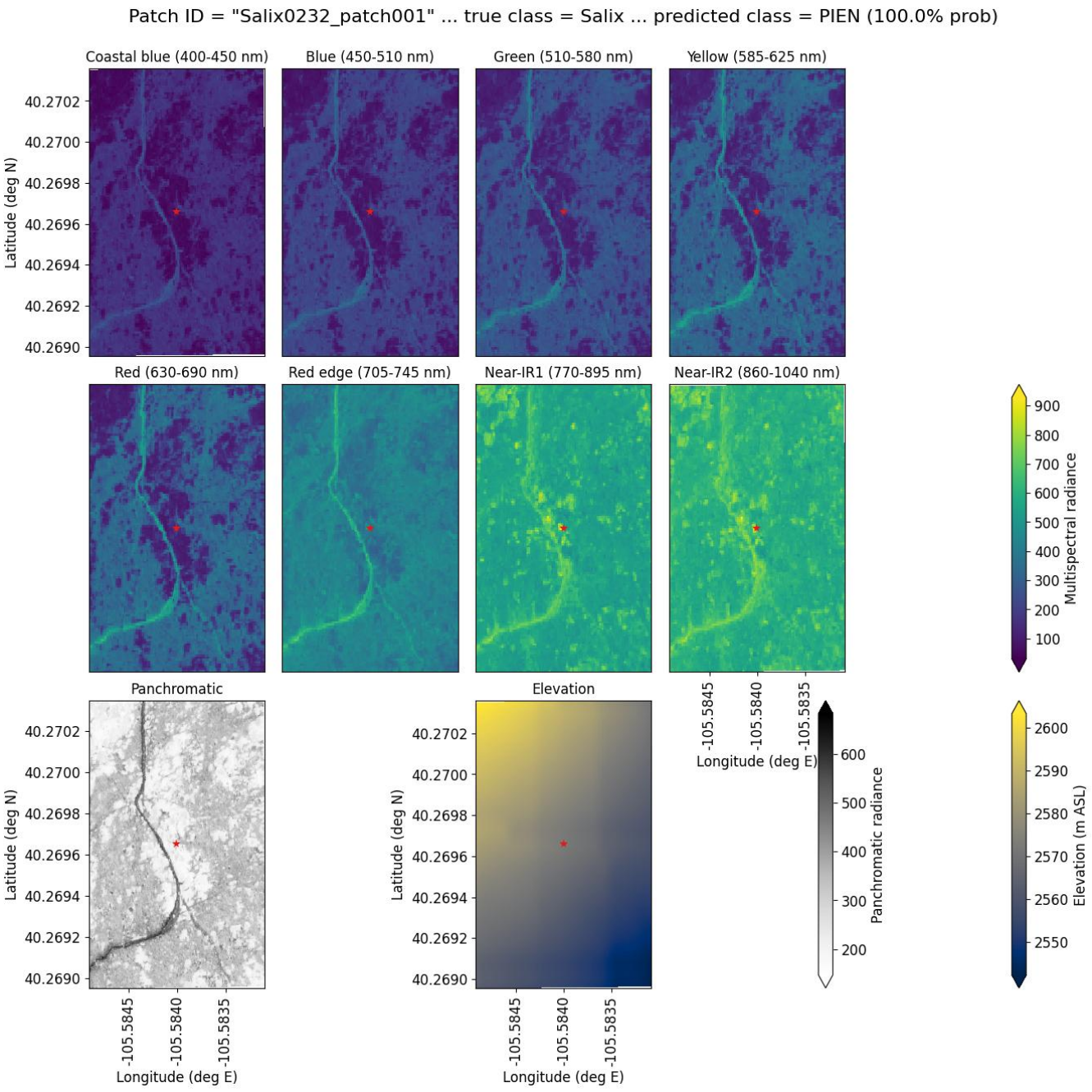


Figure S73. An example of a "worst confusion" classification of *Salix* as PIEN from the six-class model, where the model incorrectly predicted *Salix* as PIEN with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S74 is an example of worst confusion where *Salix* was the observed species and POTR was the predicted species. Both willow and aspen occur in this area, but willow is by far the most abundant, growing densely near a stream at the study site. However, the elevational and textural information are similar to other aspen examples.

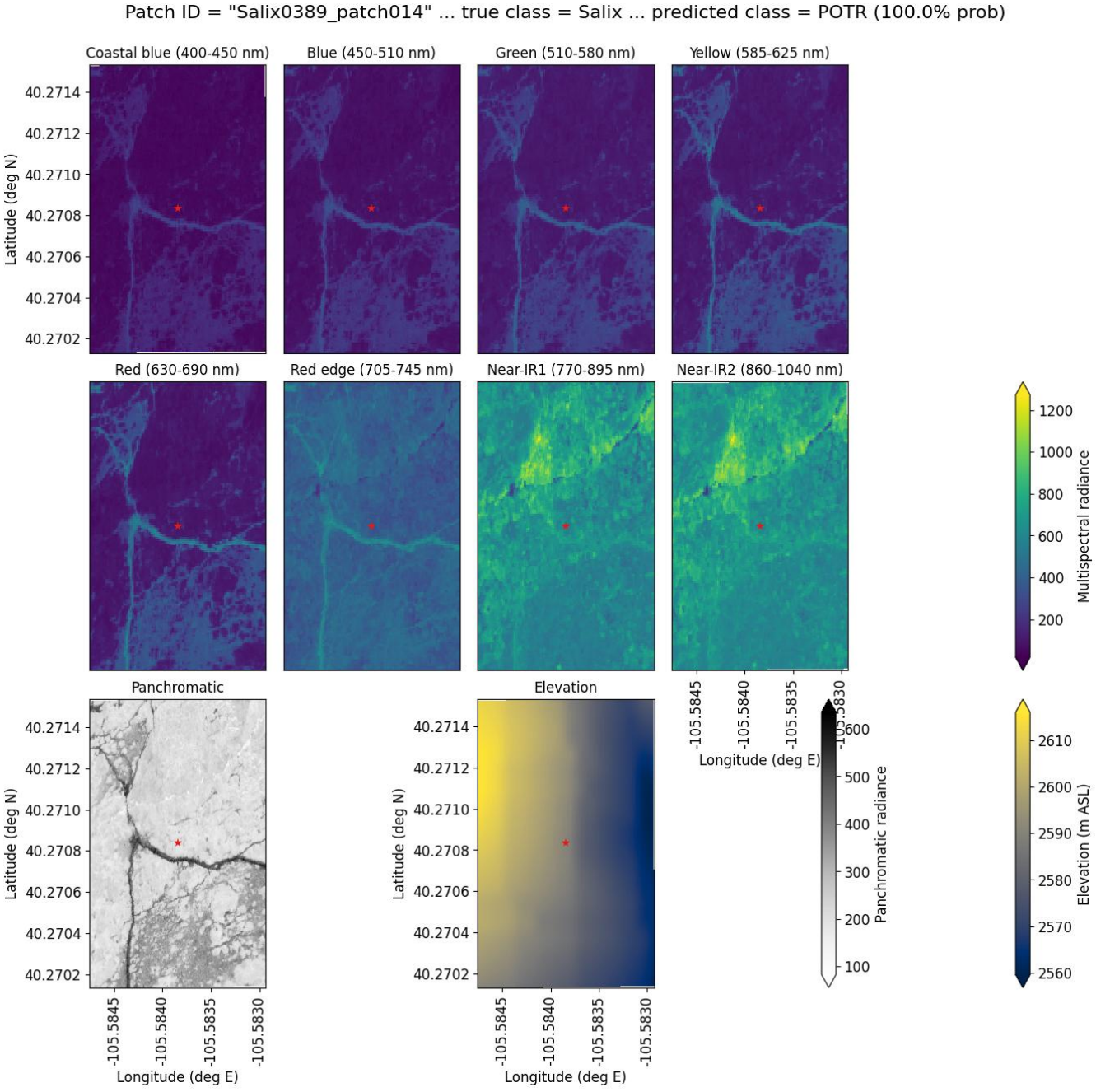


Figure S74. An example of a "worst confusion" classification of *Salix* as POTR from the six-class model, where the model incorrectly predicted *Salix* as POTR with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

Figure S75 is an example of worst confusion where BEGL was the observed species and POTR was the predicted species. This is an area where both species were found, and so the elevational and textural information/context are similar.

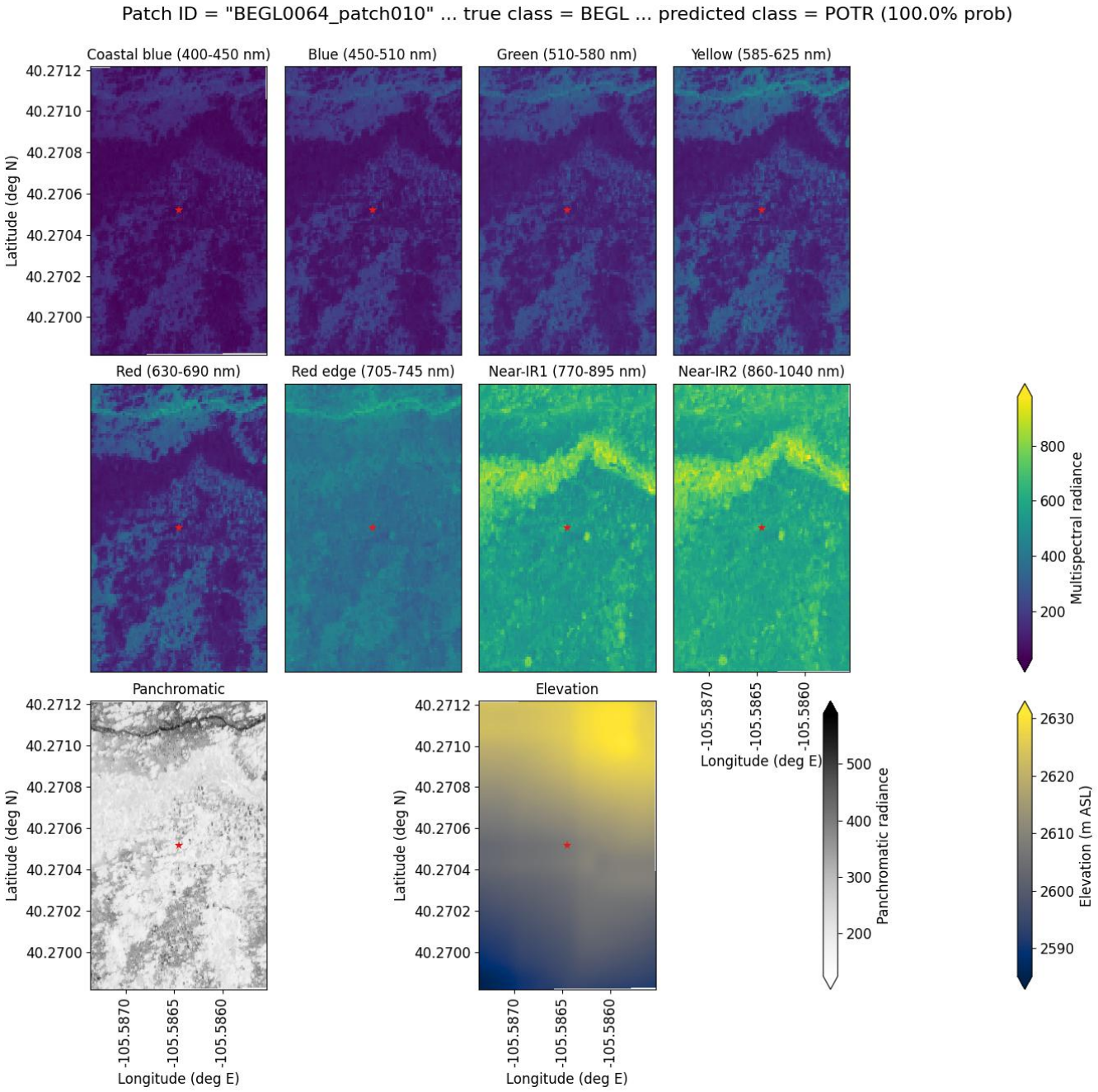


Figure S75. An example of a "worst confusion" classification of BEGL as POTR from the six-class model, where the model incorrectly predicted BEGL as POTR with 100% probability. This example (image chip or patch) is from the Longs Peak study site. Units of radiance are W m^{-2} sr^{-1} μm^{-1}.

In most cases of worst confusion, the pixel and image chip (patch) example were located in areas where the predicted species is commonly found, or where the textural and elevational information are similar to places where the predicted species is usually found. The cases of worst confusion emphasize the difficulties the model runs into by relying too heavily on the fine-scale patterns of the panchromatic imagery without the specificity of the multispectral imagery. Given that the multispectral imagery shows that pairs of species diverge in many bands (see section S3, especially Table S1), we may want to try re-training the models after resampling the panchromatic imagery. An object-based classification approach might also help improve classification accuracy. It's also possible that the multispectral data have too low a spatial resolution to be very useful for tree species classification in this system.

REFERENCES

Dubey, A. K. and Jain, V.: Comparative study of convolution neural network's Relu and Leaky-Relu activation functions, Applications of Computing, Automation and Wireless Systems in Electrical Engineering, Singapore, 2019//, 873-880,

Fukushima, K. and Miyake, S.: Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition, Competition and Cooperation in Neural Nets, Berlin, Heidelberg, 1982//, 267-285,

Glorot, X., Bordes, A., and Bengio, Y.: Deep Sparse Rectifier Neural Networks, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research2011.

Goodfellow, I., Bengio, Y., and Courville, A.: Convolutional Networks, in: Deep Learning, The MIT Press, 2016a.

Goodfellow, I., Bengio, Y., and Courville, A.: Regularization, in: Deep Learning, The MIT Press, 2016b.

Goodfellow, I., Bengio, Y., and Courville, A.: Machine Learning Basics, in: Deep Learning, The MIT Press, 2016c.

Ioffe, S. and Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Proceedings from the International Conference on Machine Learning, 448-456,

Krizhevsky, A., Sutskever, I., and Hinton, G. E.: ImageNet classification with deep convolutional neural networks, Commun. ACM, 60, 84–90, 10.1145/3065386, 2017.

Lagerquist, R.: Using deep learning to improve prediction and understanding of high-impact weather, School of Meteorology, University of Oklahoma, Norman, OK, 290 pp., 2020.

Lagerquist, R., McGovern, A., and Gagne Ii, D. J.: Deep learning for spatially explicit prediction of synoptic-scale fronts, Weather and Forecasting, 34, 1137-1160, https://doi.org/10.1175/WAF-D-18-0183.1, 2019.

Maas, A. L.: Rectifier Nonlinearities Improve Neural Network Acoustic Models,

Nair, V. and Hinton, G. E.: Rectified Linear Units Improve Restricted Boltzmann Machines, Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel,

Perez, L. and Wang, J.: The effectiveness of data augmentation in image classification using deep learning, https://doi.org/10.48550/arXiv.1712.04621, 2017.