The University of Texas at Austin
Oden Institute for Computational Engineering and Sciences

April 29, 2022

Satellite Image Classification

CSE 382M – Foundations of Data Science
Final Project

Julie Pham
PhD Student – Aerospace Engineering & Engineering Mechanics

Sebastian Henao-Garcia
PhD Student – Computational Science, Engineering, and Mathematics

Vignesh Sella
PhD Student – Computational Science, Engineering, and Mathematics

Outline

1. Introduction to the dataset and the problem
2. Dimensionality reduction of the feature space
3. Implementation of classifiers
    1. k-Nearest Neighbors (k-NN)
    2. Support Vector Machine (SVM)
    3. Convolutional Neural Network (CNN)
4. Comparison of the classifiers

Image Classification: Dataset

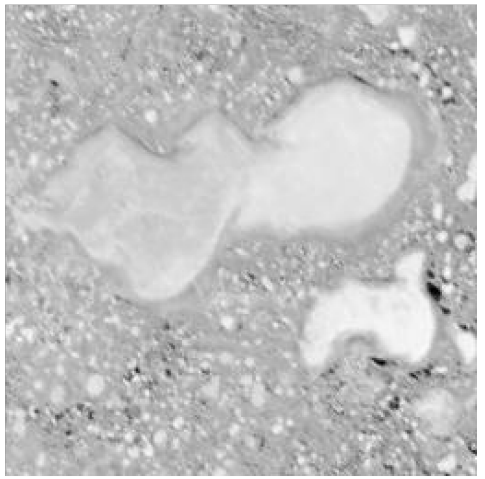
Remote **S**ensing **I**mage **S**cene **C**lassification (NWPU-RESISC45)[1] image dataset

- 256 x 256 pixel resolution, spatial resolution ~30 m to 20 cm
- Subset selection of 4 classes with 700 images each
- 60% training, 15% validation, 25% testing

Basketball court

Baseball diamond

Railway station



Lake

Dimensionality Reduction via PCA

Number of datapoints

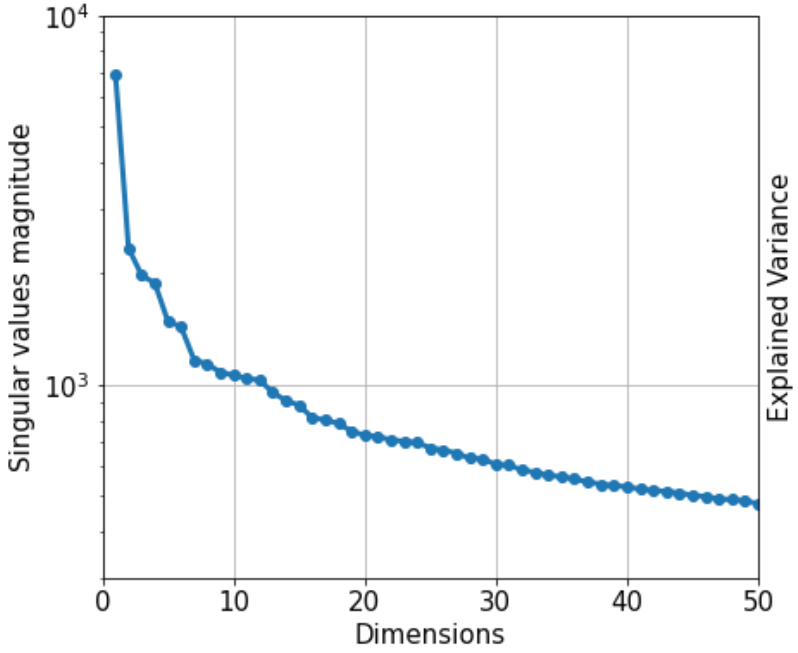
Original data

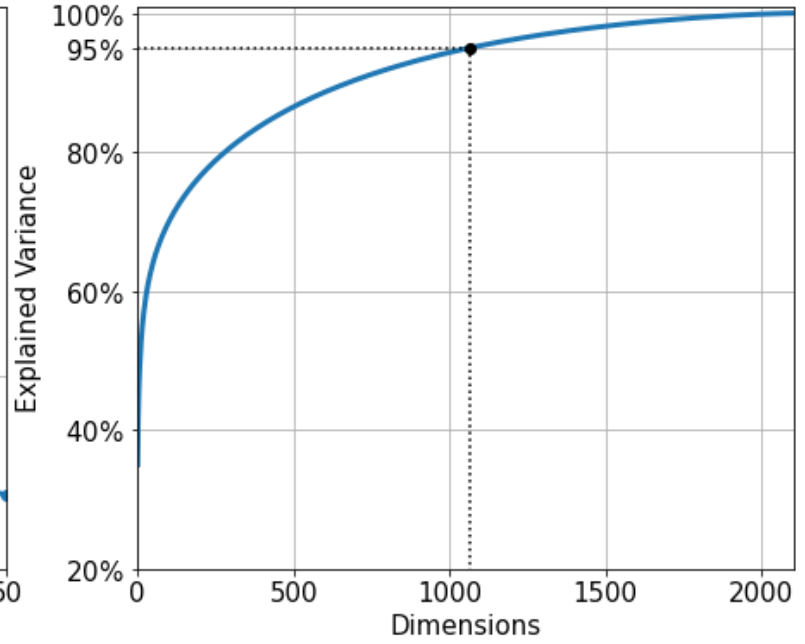
PCA

Projected data

Raw features (65,536)

Principal components





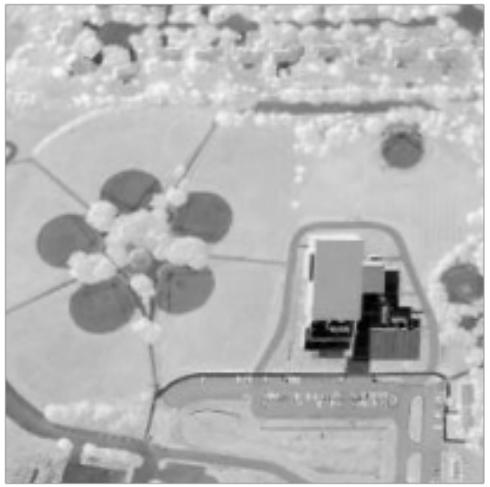
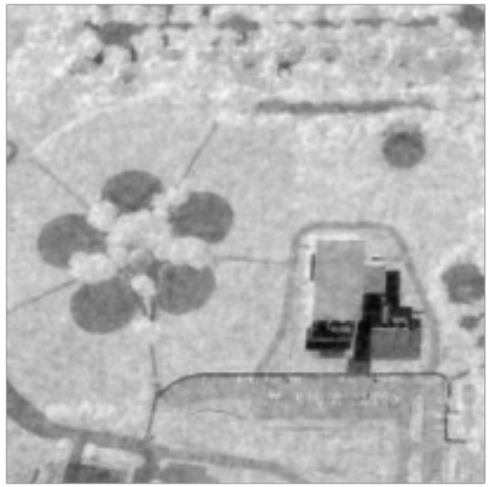
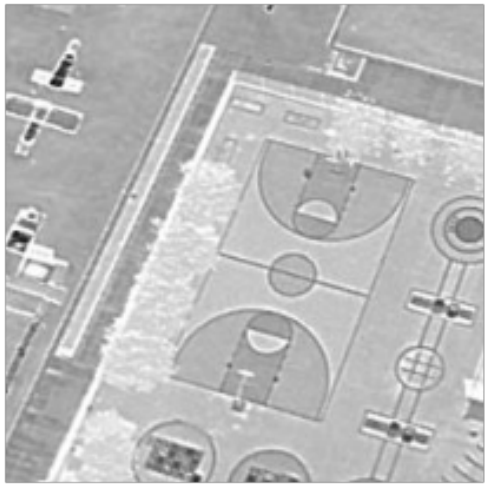
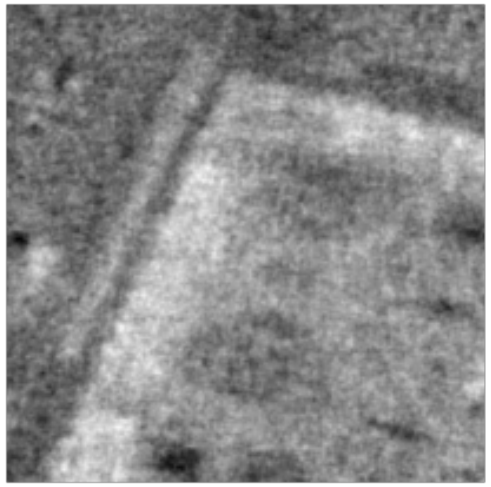
Principal components:

$$XV = U\Sigma$$

Explained variance

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2}$$

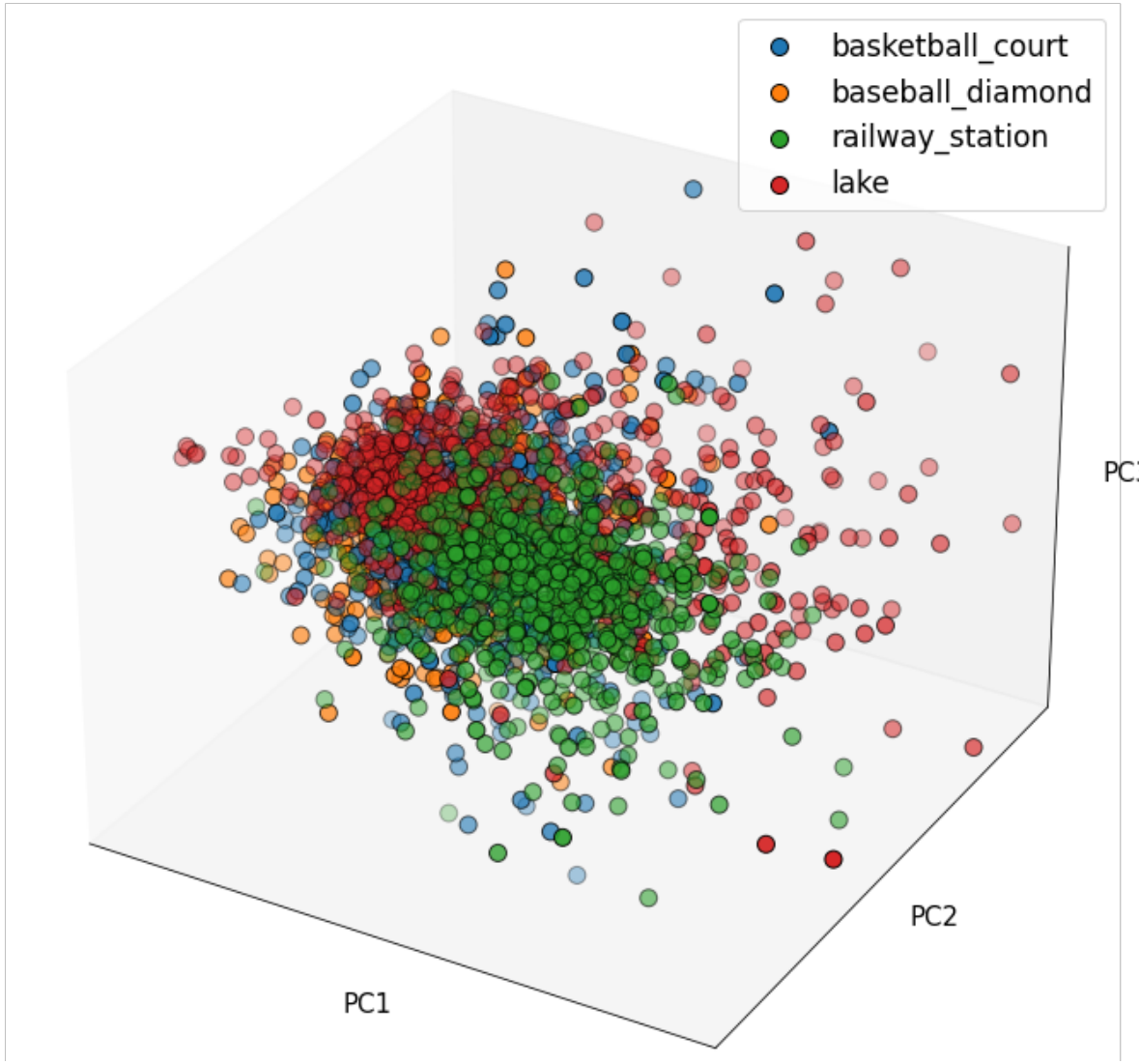
Dimensionality Reduction via PCA

	Original	Reduced
Training data		
Test data		

The PCA modes are computed using the training data set only

The test data set is then projected onto the computed PCA modes

Dimensionality Reduction via PCA



Using only 3 reduced features, we see some separation in the data, but it is still not strong enough to draw a clear decision boundary.

Then, we need to include more features, but the optimal number depends on the nature of the classification algorithm to be trained.

Our goal

To train three different machine learning models to predict the class a given image belongs to based on:

- k Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Convolutional Neural Network (CNN)

Comparing them through:

- Prediction accuracy
- Misclassifications among classes

k-NN algorithm

1. Compute (Euclidean) distance between query point and all points in training dataset
2. Select k points with the smallest distances to query point
3. Obtain labels corresponding to the k points
4. Count how many of each label there are
5. Predict the mode

k-NN: Curse of Dimensionality

What happens when your data is high-dimensional?

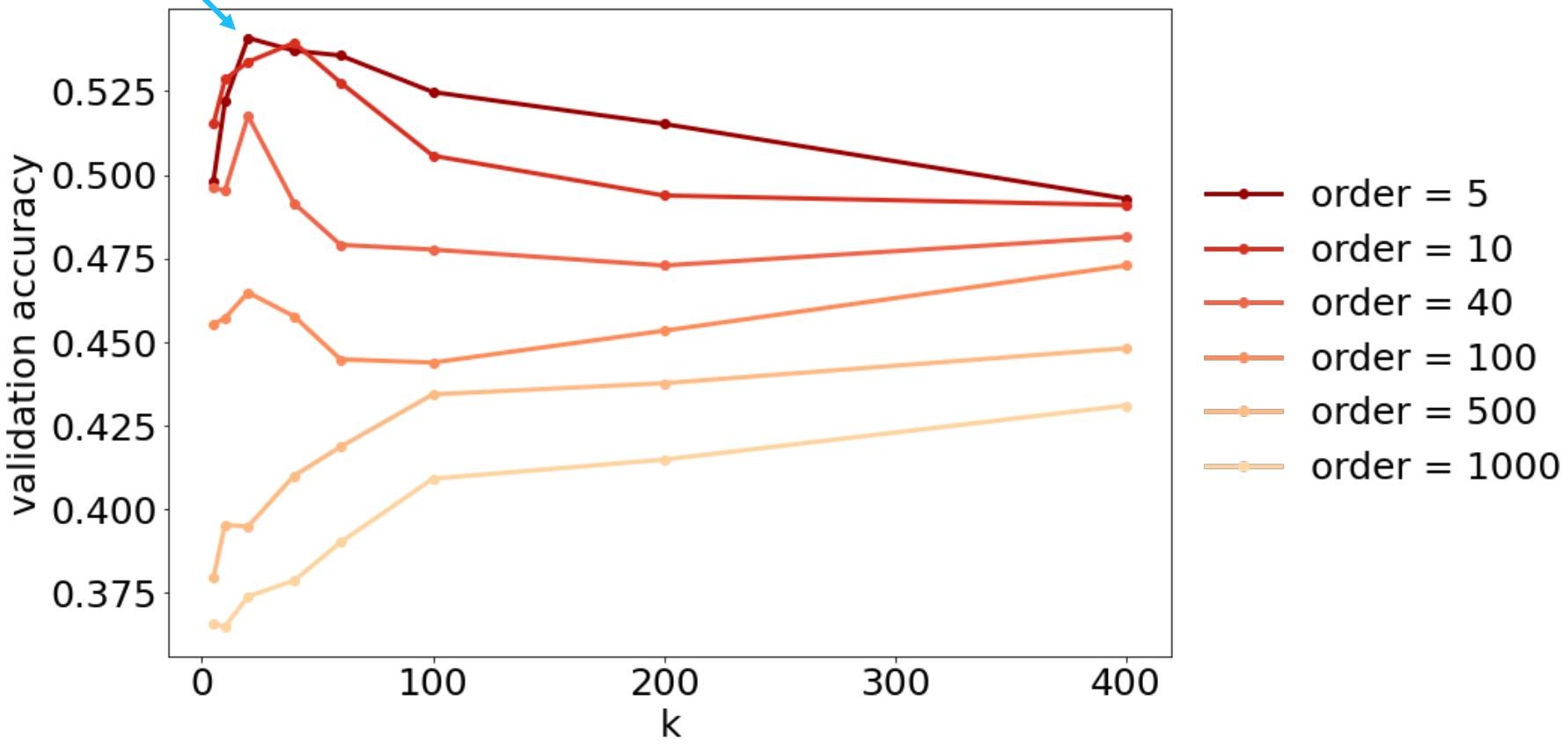
…the Euclidean distances look equally dissimilar!

Dimension reduction is necessary in this case,
but centering/scaling does not make a big difference.

k-NN: Hyperparameter tuning & test accuracy

Optimal parameters:
r = 5, k = 20

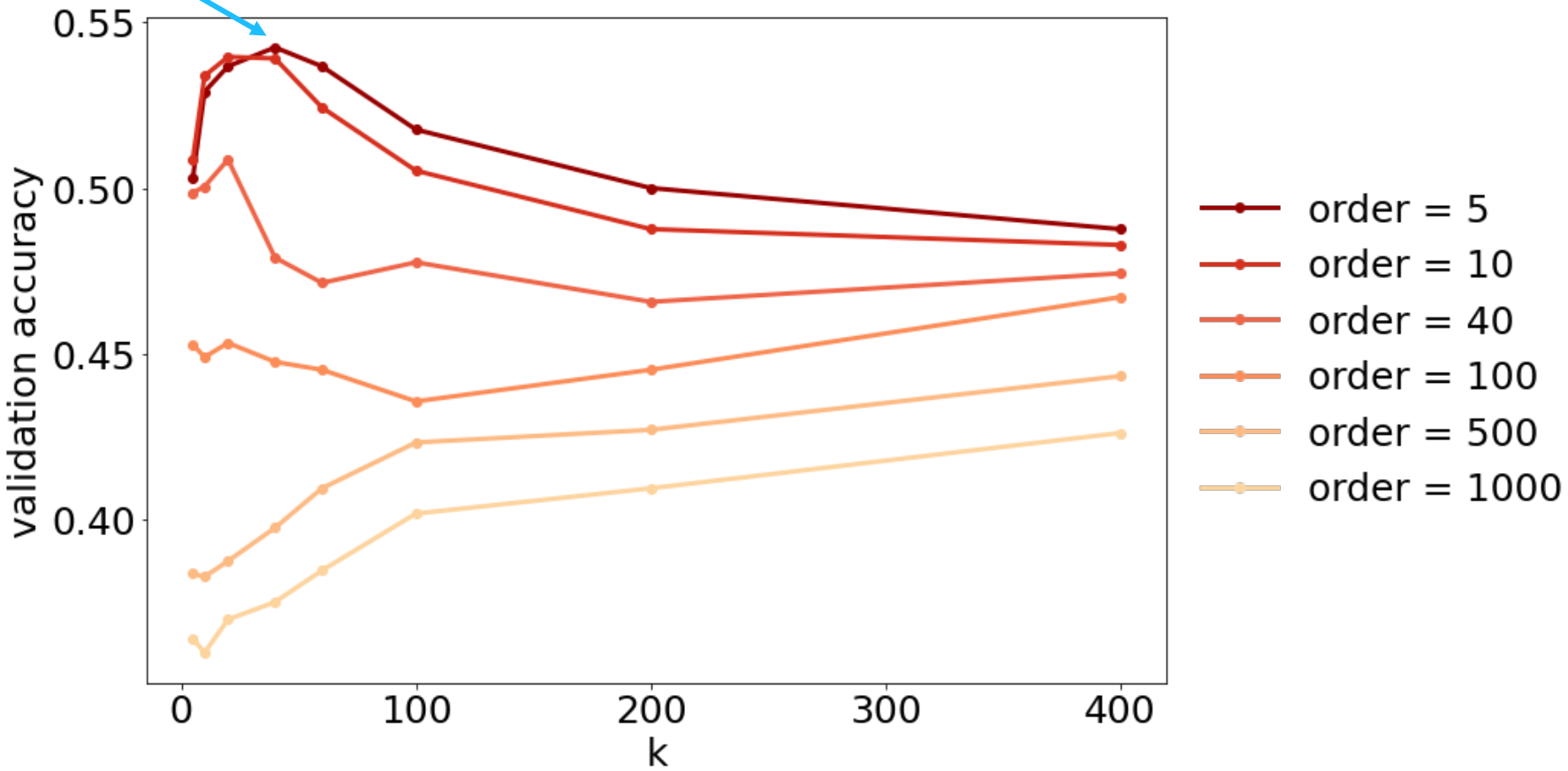
Testing accuracy = 51.3%



k-NN: Add centering/scaling

Optimal parameters:
r = 5, k = 40

Testing accuracy = 53.3%



SVM: Problem setup

Goal: To find an optimal separation boundary **(a hyperplane)** between two classes such that the distance between the **support vectors** of the classes **(the margin)** is maximized.

Primal form: $\min_{\mathbf{w},b,\xi} \|\mathbf{w}\|_2^2 + C\sum_{j=1}^{n}\xi_j$ such that $y_j \cdot (\langle \mathbf{w}, \mathbf{x}_j \rangle - b) \geq 1 - \xi_j, \quad \xi_j \geq 0$

Weights vector normal to the hyperplane

Allows for misclassification

Important:

- This is problem is formulated as binary classification.
- SVM is not scale invariant, therefore data needs to be standardized before training the classifier.

Rewriting the optimization problem in its **dual form** through Lagrange multipliers, allows us **to use the kernel trick** to include **nonlinearity in the hyperplane**

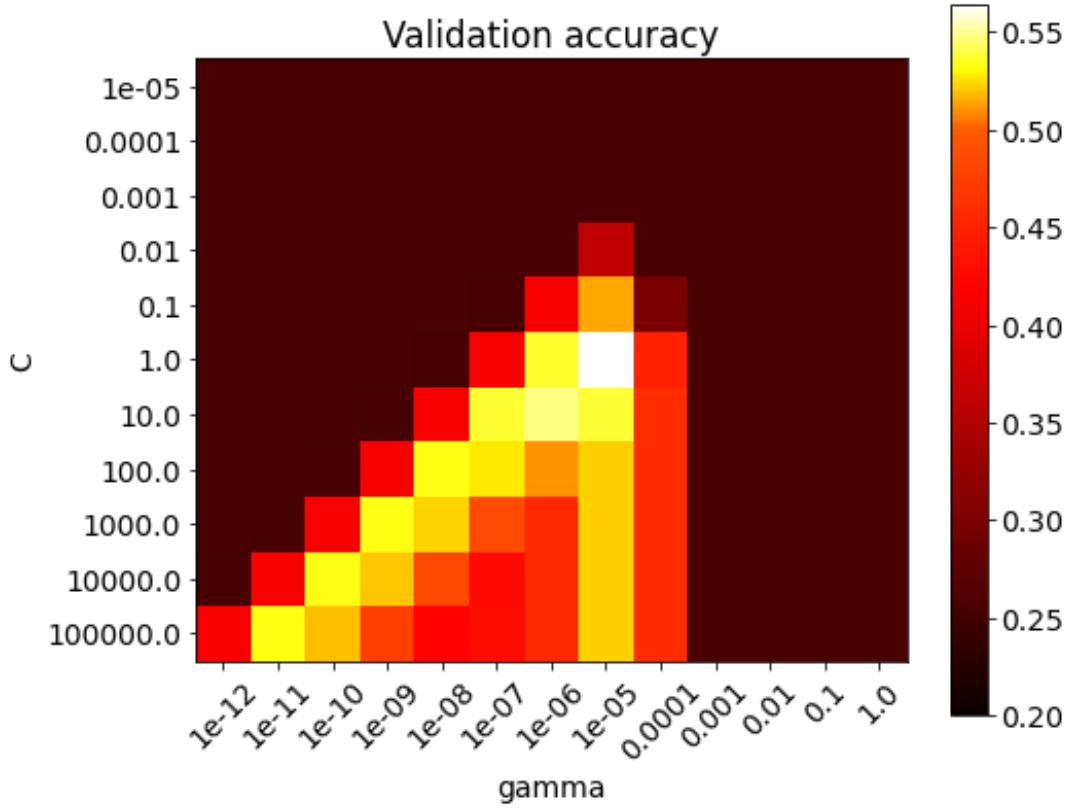
Evaluated kernels:

$$\text{Linear:} \quad \langle x, x' \rangle$$
$$\text{Polynomial:} \quad \left(\gamma \langle x, x' \rangle + r\right)^d$$
$$\text{Sigmoid:} \quad \tanh\left(\gamma \langle x, x' \rangle + r\right)$$
$$\text{RBF:} \quad \exp\left(\gamma \|x - x'\|\right)$$

SVM: Implementation

- Trained with enough PCA modes to capture 95% of the energy in the training dataset: **1,061 features**.
- **Data was standardized** to have zero mean and unit variance.
- Multi-class classification implemented through **one-vs-one approach.**
- **Hyperparameter tuning** strategy:
  - Logarithmic grid to assess parameters spanning multiple orders of magnitude.
  - Training and validation datasets were combined and split into validation and training portions in 5 different ways for cross validation.
  - Optimal set of parameters obtained averaging the parameters yielding the highest accuracy for each split.

Heat map of hyperparameters logarithmic grid for RBF kernel



SVM: Results

Training with the **reduced feature space**

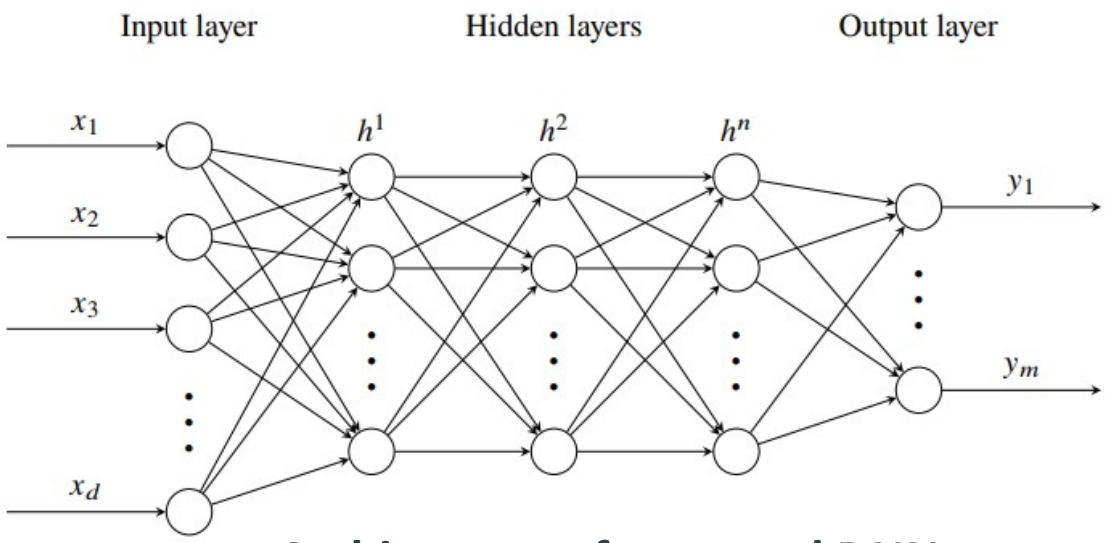
Kernel	Accuracy	
	Training	Test
Polynomial (degree 3)	98%	47%
Linear	66%	54%
Sigmoid	66%	54%
RBF	77%	56%

Training with the **raw feature space** using the same optimal hyperparameters

Kernel	Accuracy	
	Training	Test
Polynomial (degree 3)	100%	47%
Linear	68%	54%
Sigmoid	68%	54%
RBF	80%	56%

Deep Neural Networks (DNN)

- DNNs train neurons with weights and non-linear activation functions, grouped in layers, to minimize a loss function and predict on new data
- DNNs inherently scale poorly to highly dimensional data
  - e.g., a 256 x 256 pixel resolution image would require 65536 weights per neuron in dense layer
- We use a class of DNNs, referred to as convolutional neural networks (CNNs)

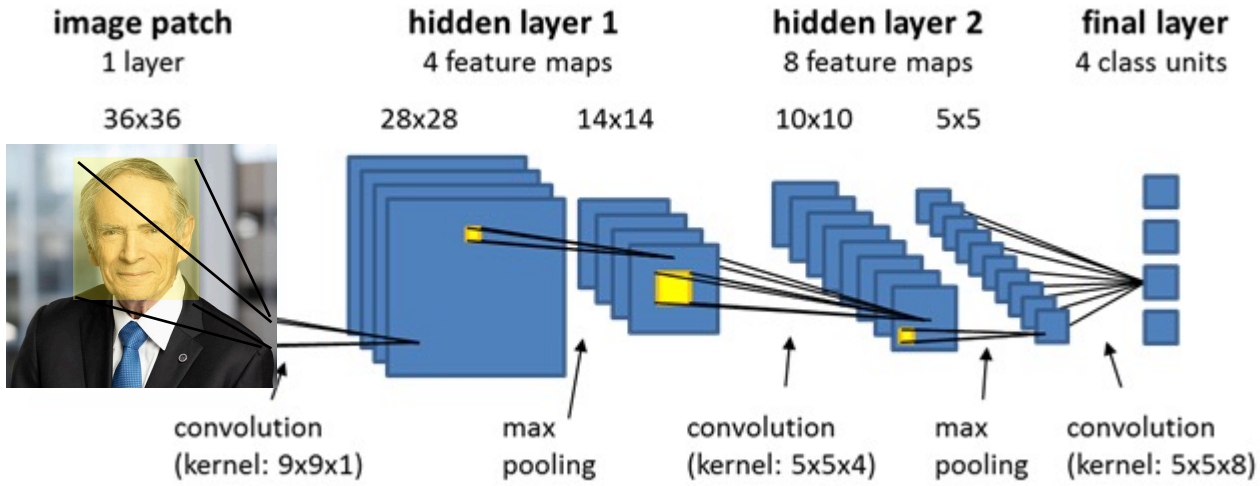


Architecture of a general DNN

Convolutional Neural Networks (CNNs)

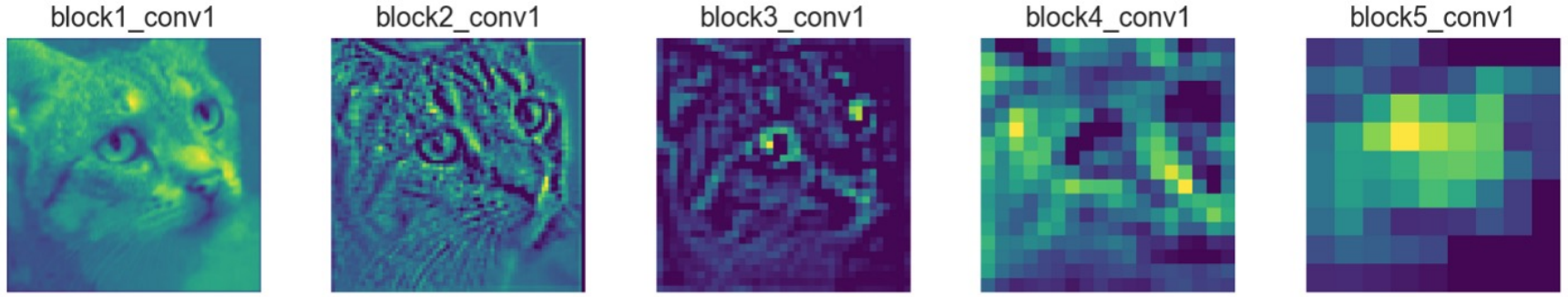
- Convolutional layers:
  - extract features and learn them at the same time
  - use significantly less parameters than a fully connected layer

- Input: (# of images) x (image height) x (image width) x (image channels)
- Output: (# of images) x (map height) x (map width) x (map channels)



Architecture of a CNN

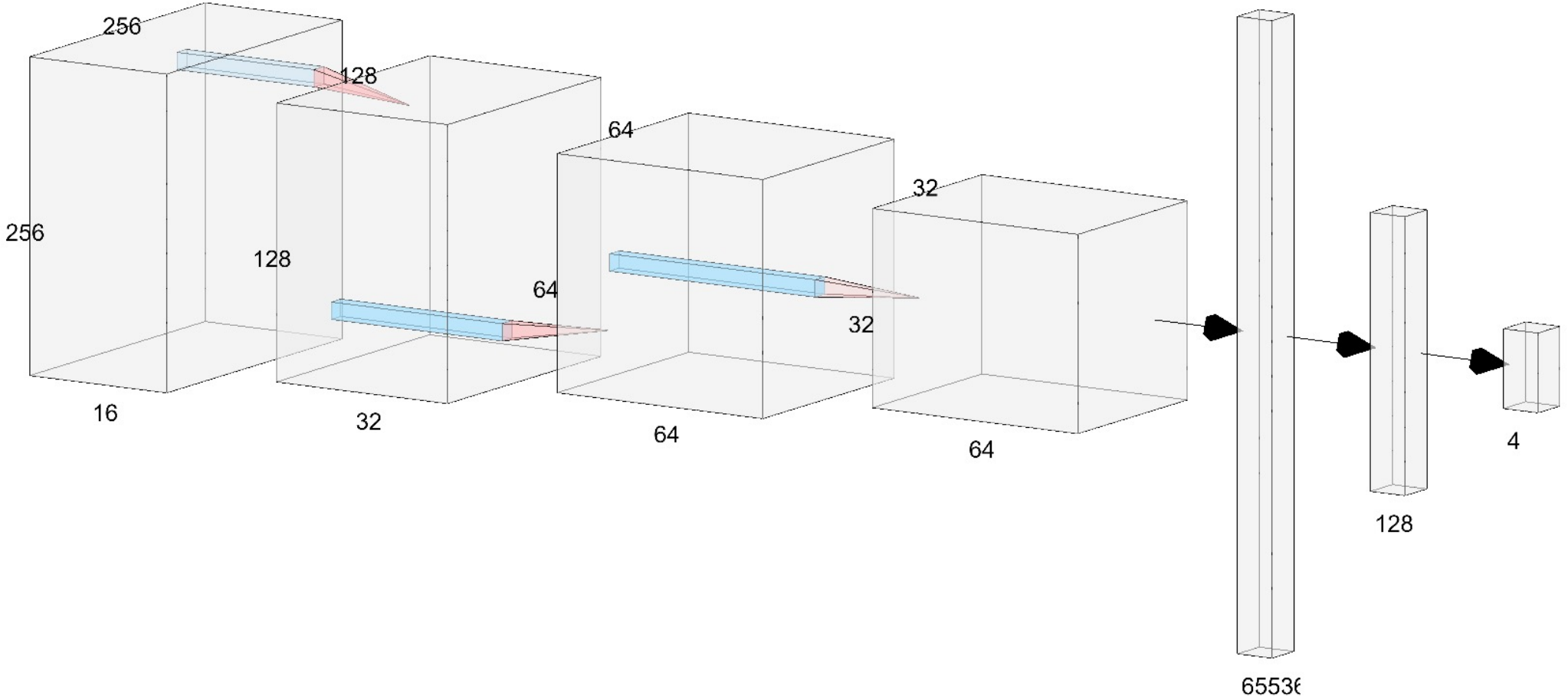
[2,3]



One feature map per layer in a CNN

[4]

CNN model architecture



CNN implementation

Training

- Data preprocessing:
  - Each batch had rescaling, a random chance of vertical flips, rotations, shears, width shifts, height shifts, zooming, and horizontal flips
- Optimizer: Adam (α = 1e-3)
- Loss: Categorical cross-entropy
- Callbacks:
  - Learning rate scheduler
  - Early stopping

Network

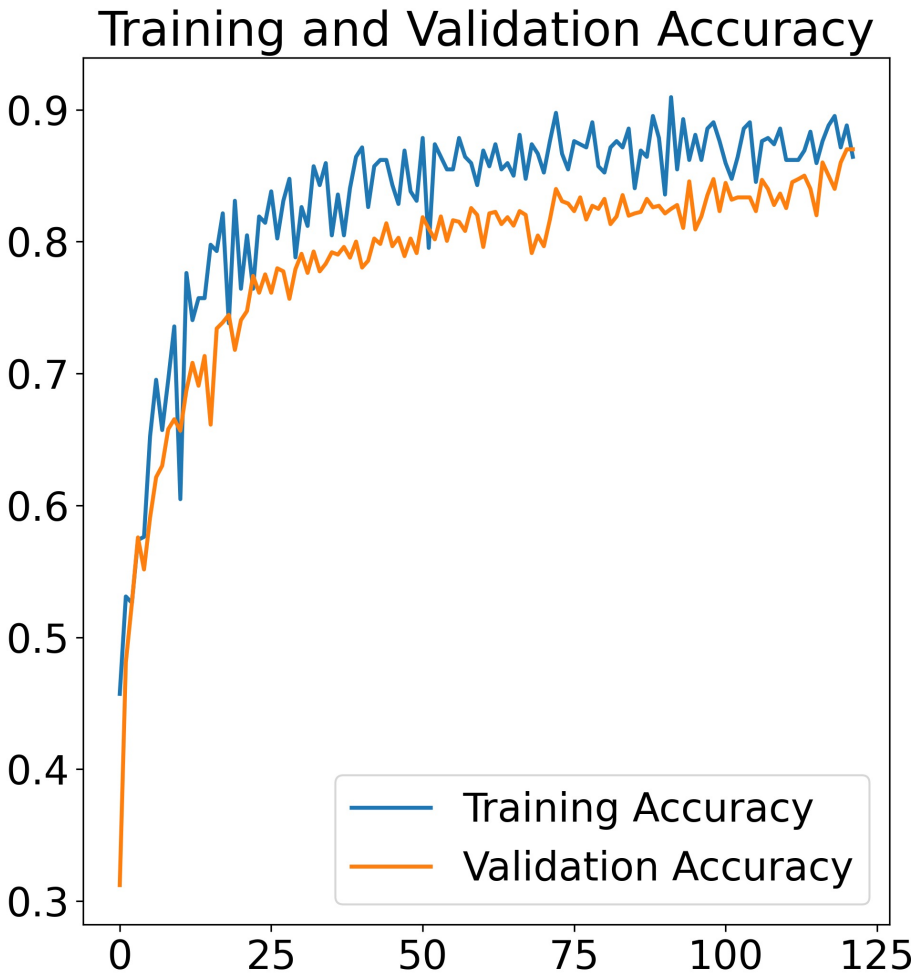
- Activation Function: ReLU
- Dropout layers (at 20%)
- L2 Kernel Regularization (λ = 1e-3)

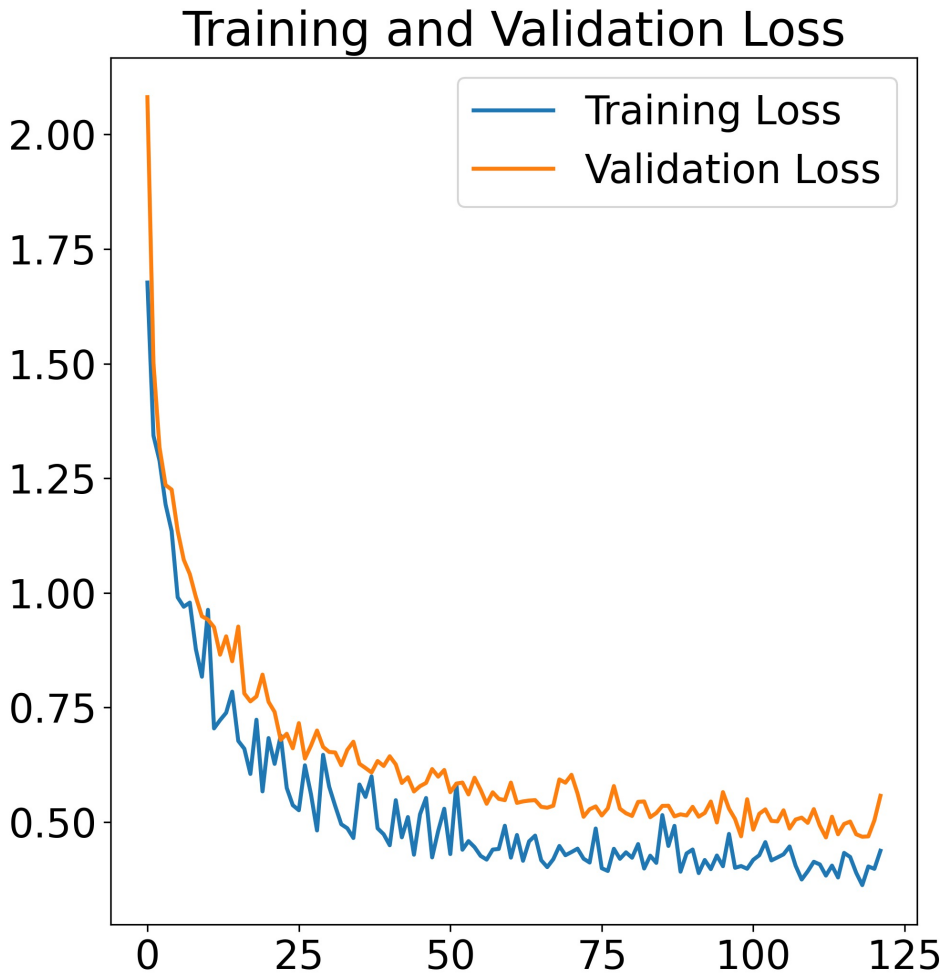
CNN in the reduced space

- Apply a CNN to projected inputs via PCA
- Attempt same network architecture, with new hyperparameters
- Experiment with variants that fit new data
- Use similar implementation tools as the CNN in full dimensional space

CNN training history

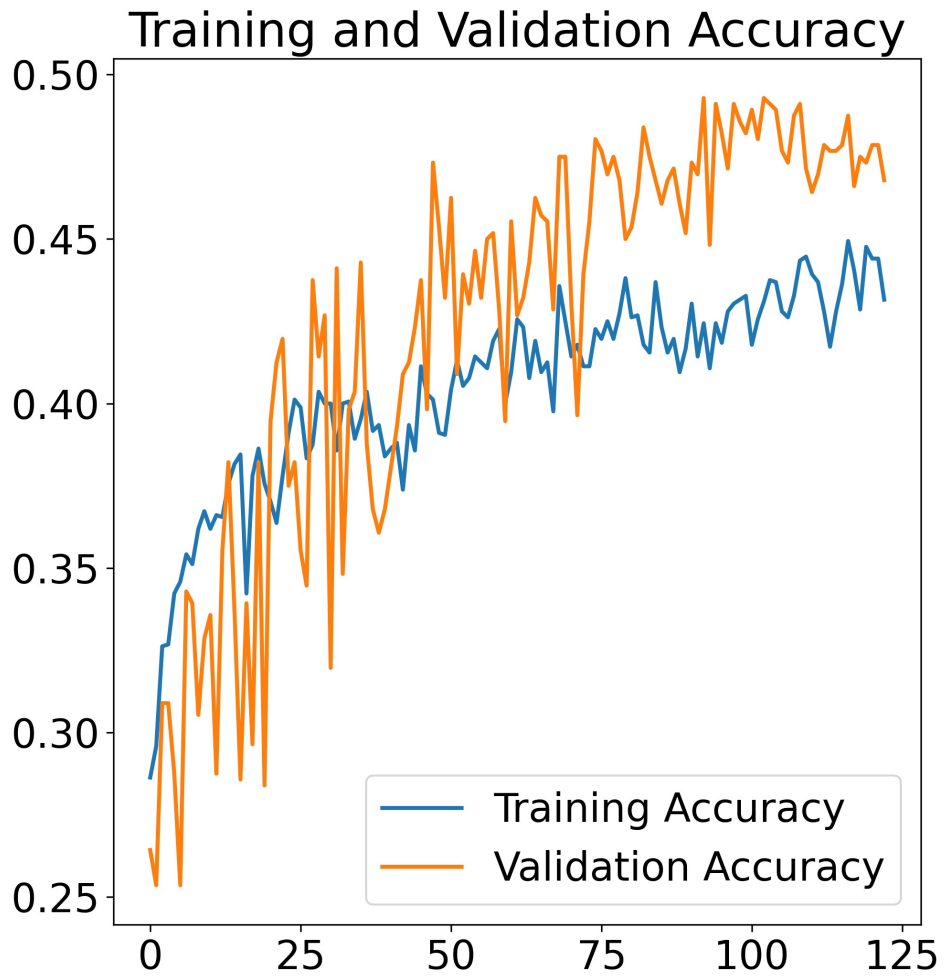
Testing accuracy = 87.0%

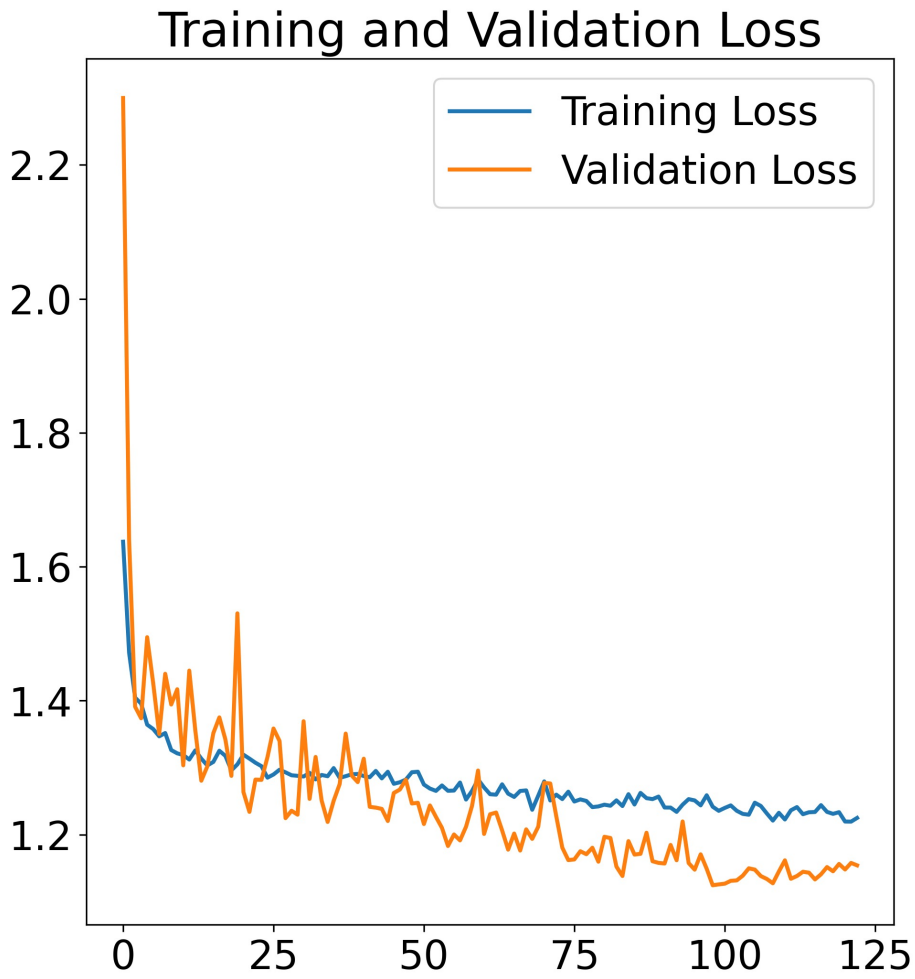




CNN training history in reduced space

Testing accuracy = 47.3%

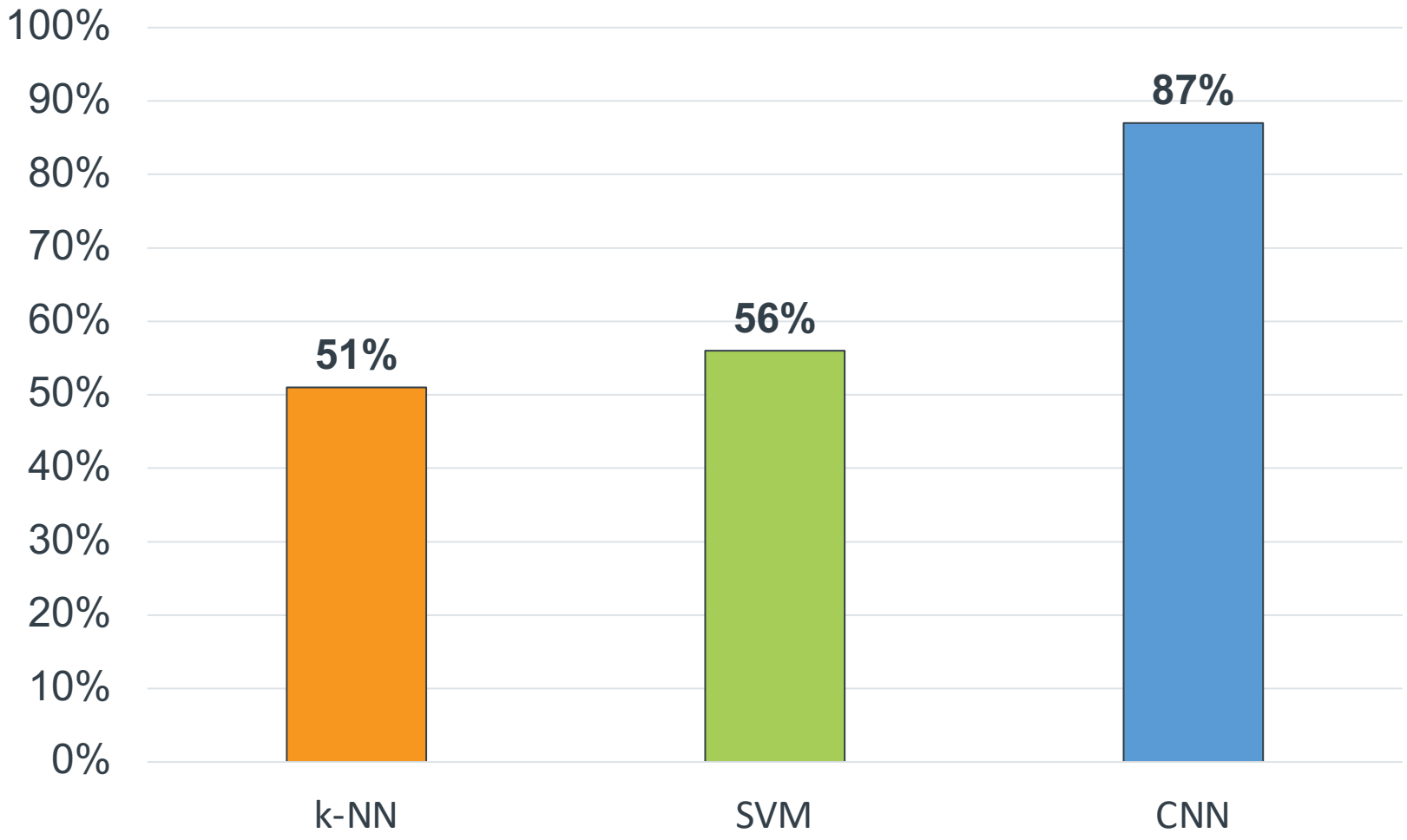




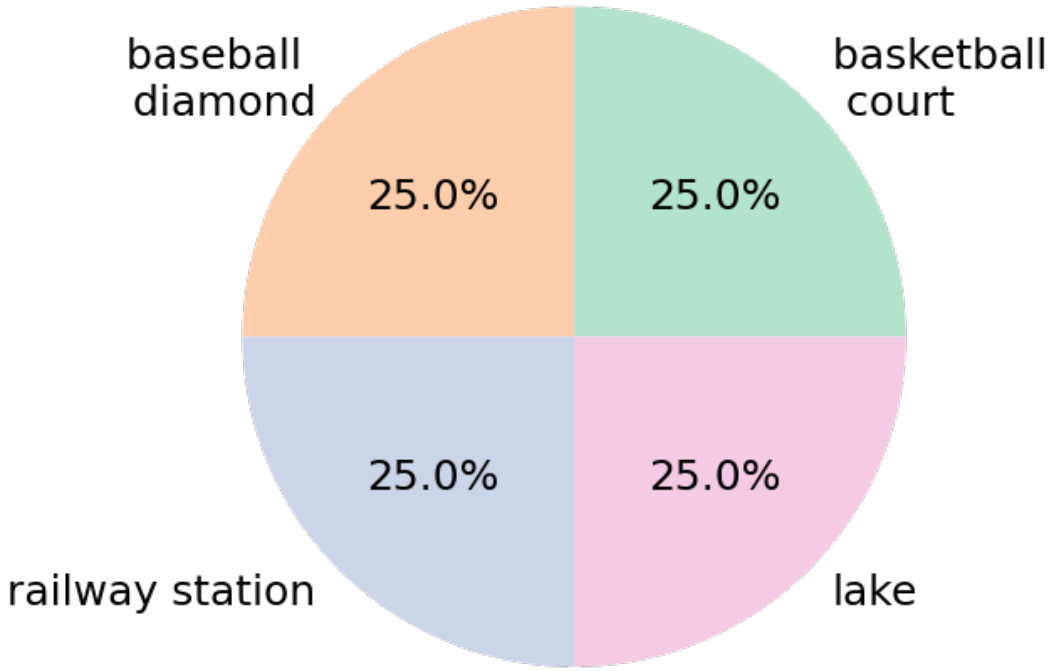
Applying PCA in context

1. **PCA ill-advised in context,**
    1. PCA has implications on the correlations and dimensionality of the data
    2. CNNs are well equipped to leverage high dimensional data and highly correlated variables
2. **CNN unable to generalize to test data, given training on training feature space/principal components**

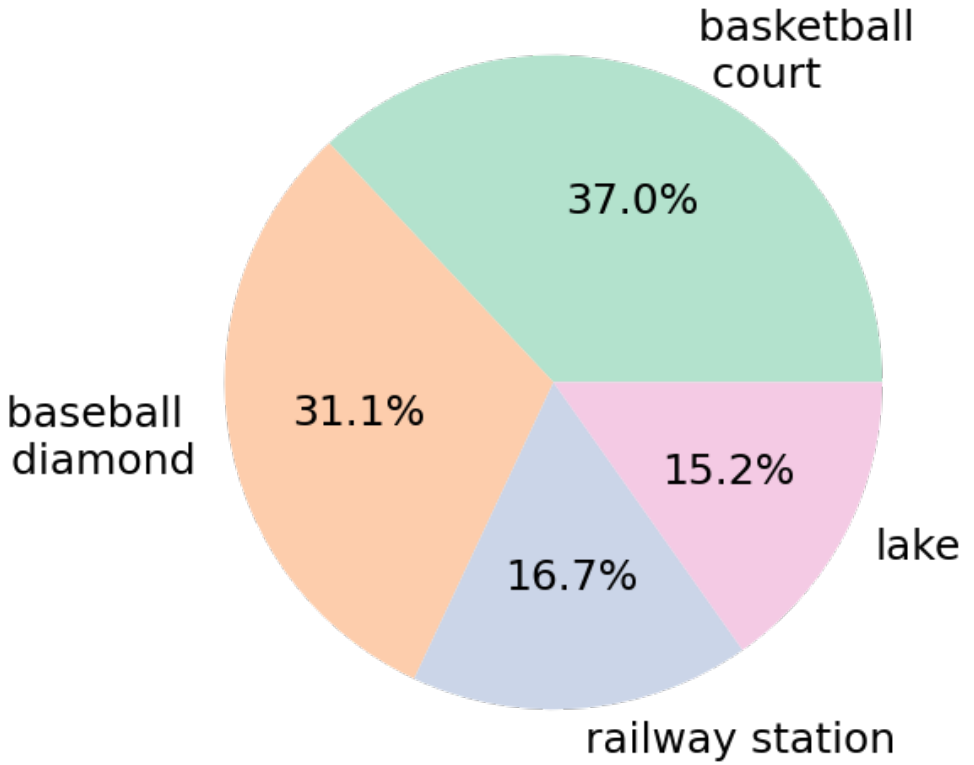
Comparing the test accuracy



k-NN Misclassifications



Training/Testing datasets Composition

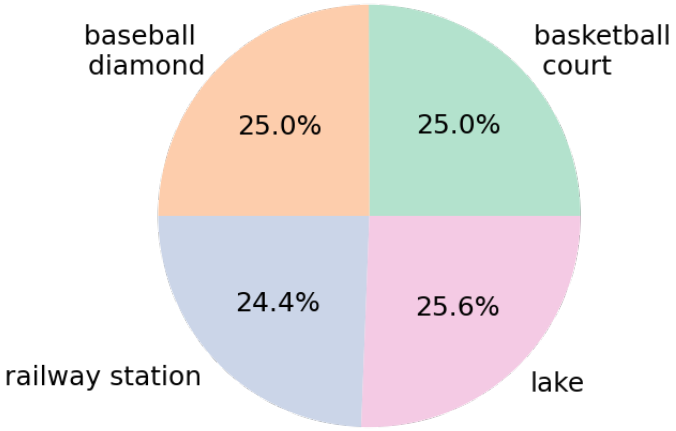


Percent of misclassifications corresponding to each class

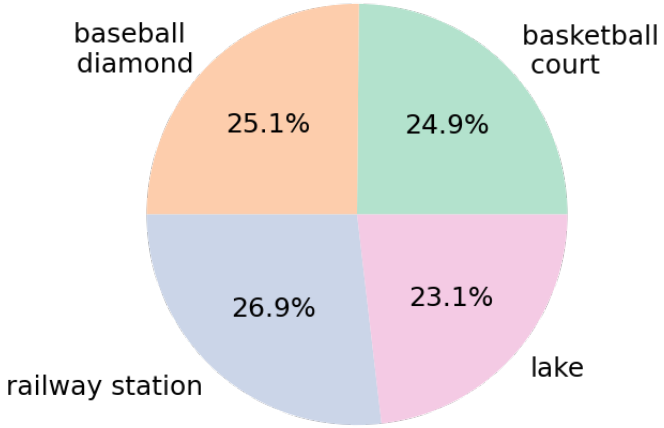
SVM Misclassifications

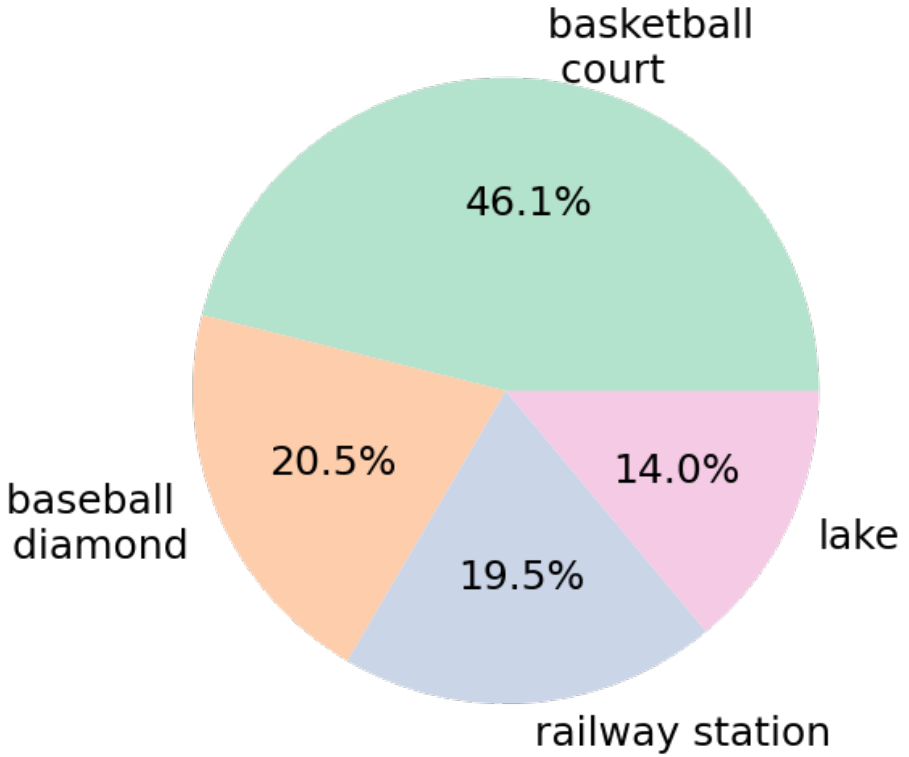
Dataset composition

Train dataset



Test dataset



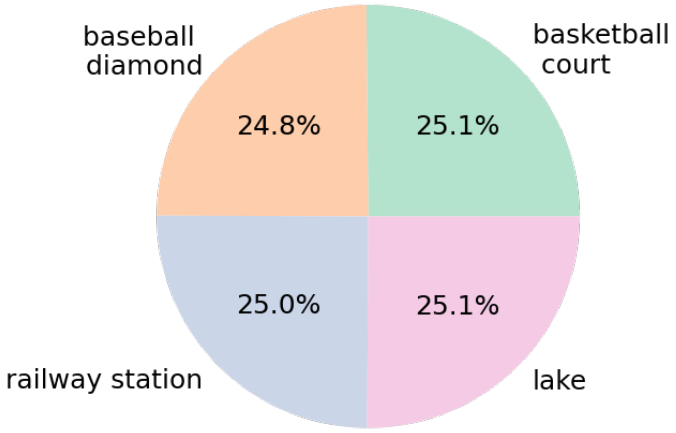


Percent of misclassifications corresponding to each class

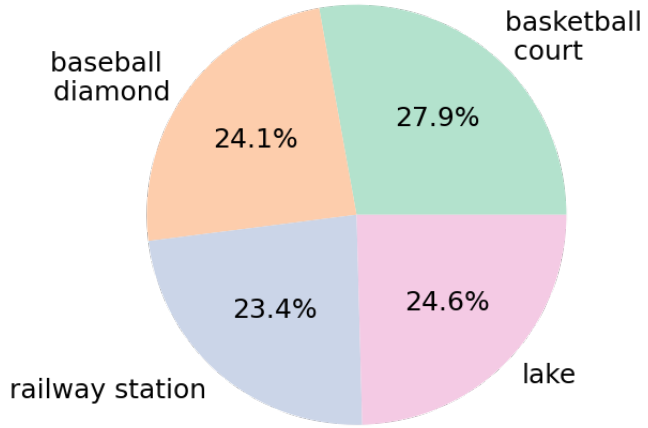
CNN Misclassifications

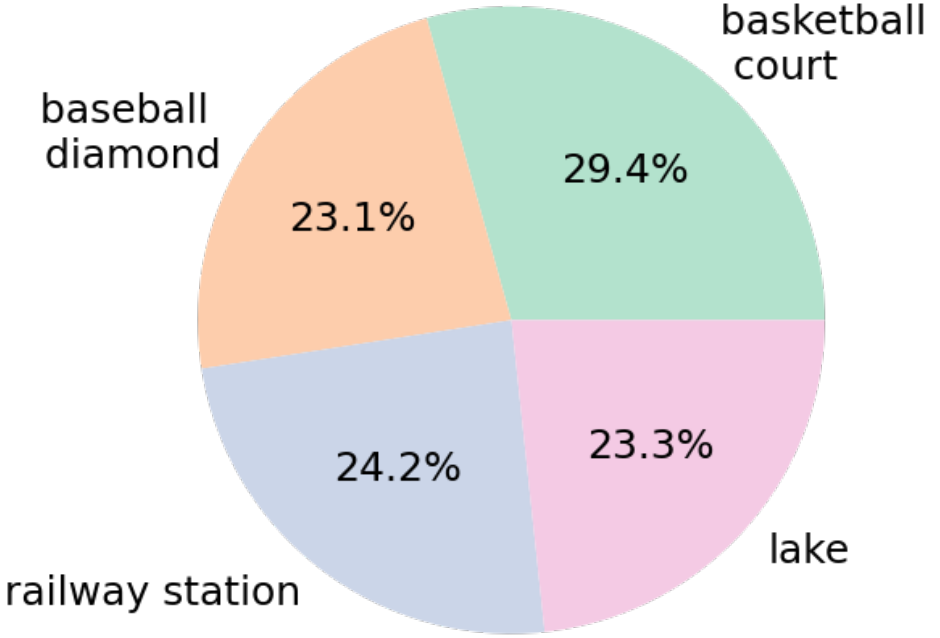
Dataset composition

Train dataset



Test dataset

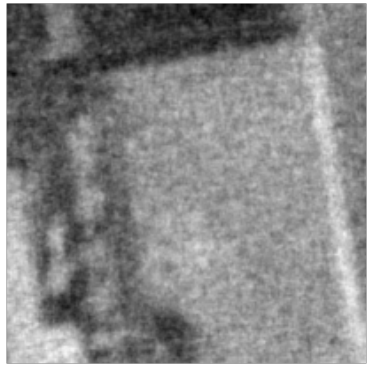




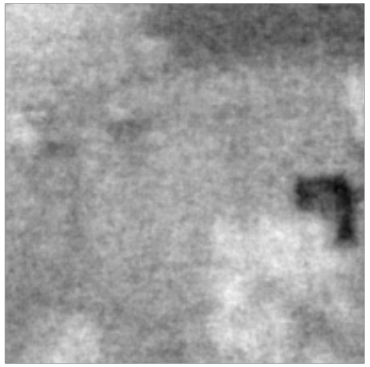
Percent of misclassifications corresponding to each class

Let's try classifying some images ourselves

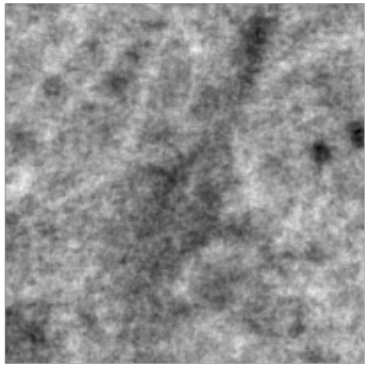
Possible classes: Basketball court, baseball diamond, railway station, lake



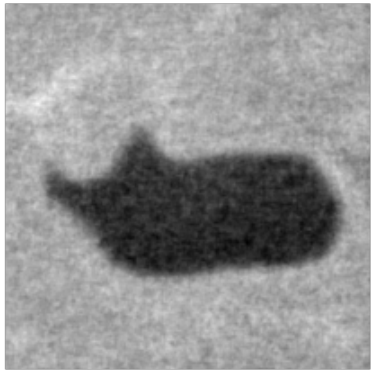
Basketball court



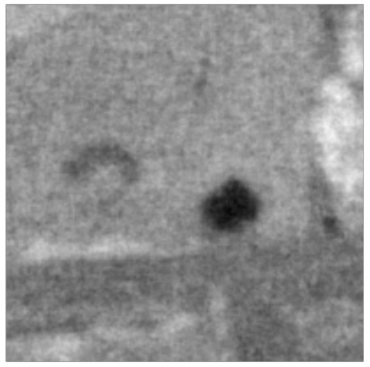
Basketball court



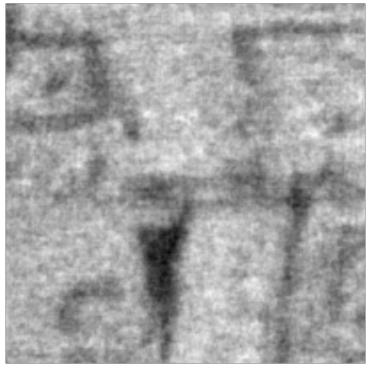
Railway station



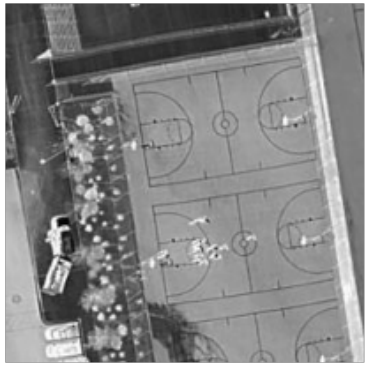
Lake?

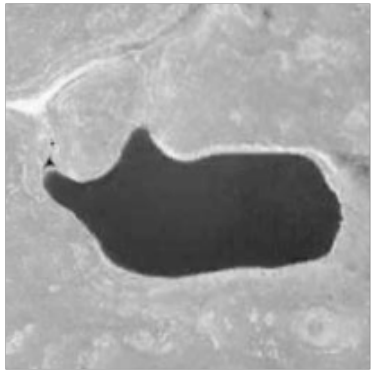


Baseball diamond

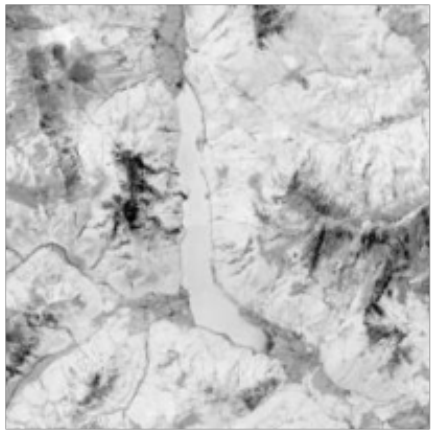
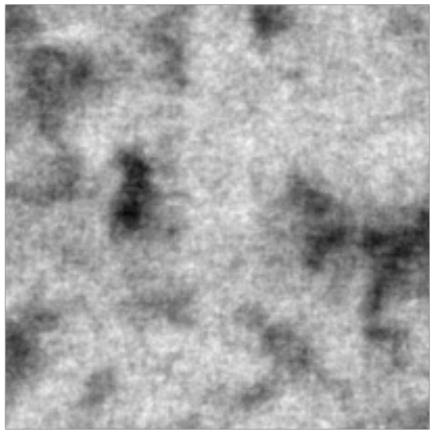
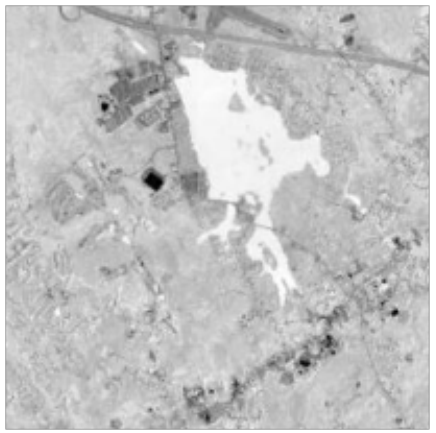
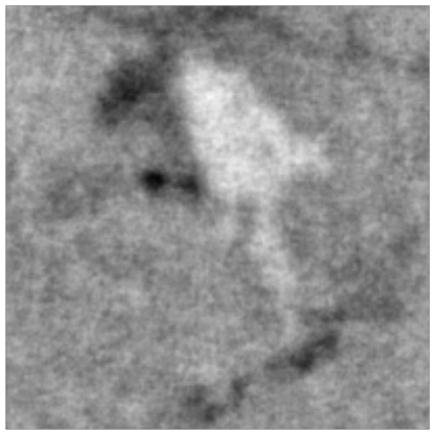


Baseball diamond

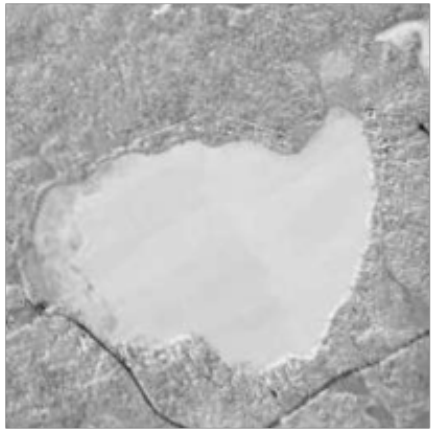
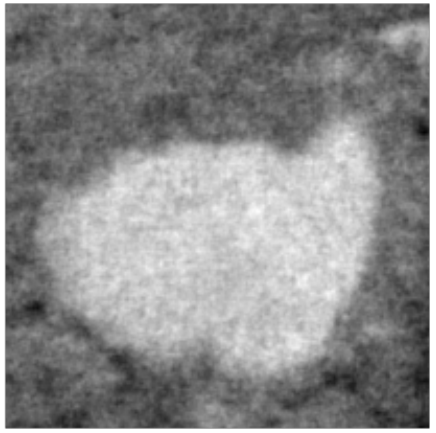
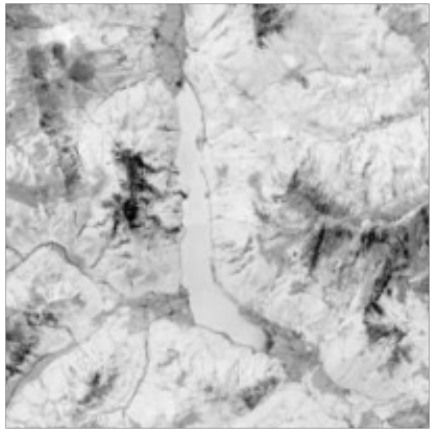
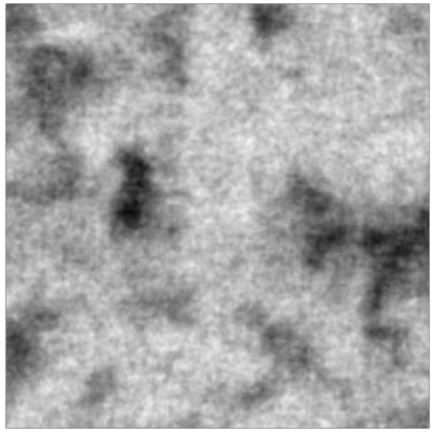




Let’s look at some lakes a little closer

Original	Reduced	Original	Reduced
			

The features defining a lake are generally stronger than for the rest of classes!

References

1. G. Cheng, J. Han and X. Lu, "Remote Sensing Image Scene Classification: Benchmark and State of the Art," in Proceedings of the IEEE, vol. 105, no. 10, pp. 1865-1883, Oct. 2017, doi: 10.1109/JPROC.2017.2675998

2. https://www.oden.utexas.edu/media/directory-assets/profile-imgs/oden.jpg

3. https://docs.ecognition.com/v9.5.0/Resources/Images/ECogUsr/UG_CNN_scheme.png

4. https://miro.medium.com/max/1400/1*OuxhgVj1WDDfo5UO5GIhgA@2x.png

The University of Texas at Austin
Oden Institute for Computational Engineering and Sciences