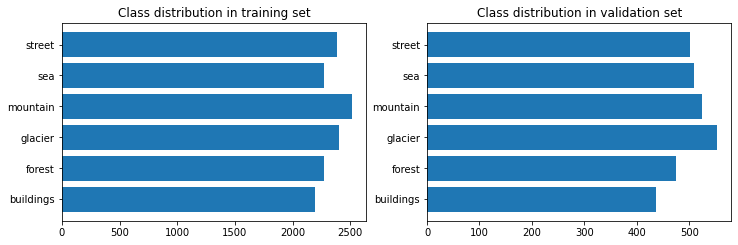
Jon Rippe  
CSCE A415 – Machine Learning  
HW #4 – Convolutional Neural Networks  
24 April 2021

# Convolutional Neural Networks & Image Recognition

## Intel Image Classification

This dataset consists of 14,034 training images and 3,000 validation images of size 150 x 150 falling into six classifications: Buildings, Forest, Glacier, Mountain, Sea, and Street. Class counts are not perfectly even within the dataset, but each class is well represented:



## Goal

Create an **efficient** CNN to classify images. Final CNN should run in the same amount of time as the Sigmoid and Relu tests shown below using the baseline model (approx. 60-70s per epoch) and should minimize overfitting. For the sake of runtime, a max of 15 epochs will be tested per implementation.

Baseline model:

Baseline batch size is 32.

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

rescaling (Rescaling) (None, 150, 150, 3) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d (Conv2D) (None, 148, 148, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 74, 74, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 175232) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 6) 1051398

=================================================================

Total params: 1,052,294

Trainable params: 1,052,294

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rescale first to change RGB values from 0-255 to 0-1 for proper weighting.

Apply one convolutional layer with 32 filters and a 3x3 kernel.

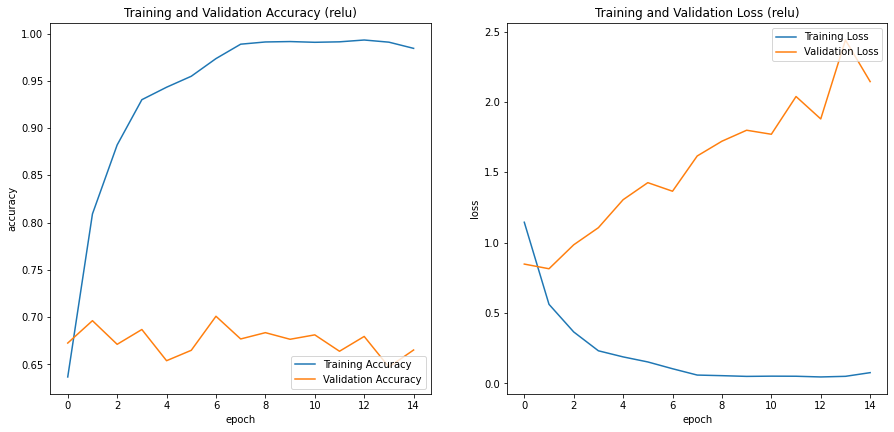
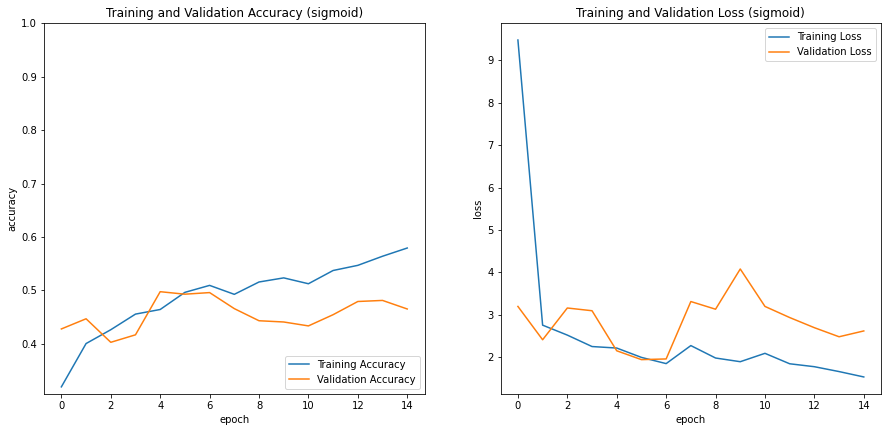
Apply one 2x2 max pool layer to reduce parameters.

Flatten and run through a final softmax layer to predict class.

## Testing Hyperparameters

### Sigmoid vs Relu

|  |  |  |
| --- | --- | --- |
| **Activation Function** | Sigmoid | Relu |
| Runtime per epoch | 62s | 58s |
| Training accuracy | ~57% | ~99% |
| Validation accuracy | ~47% | ~66% |

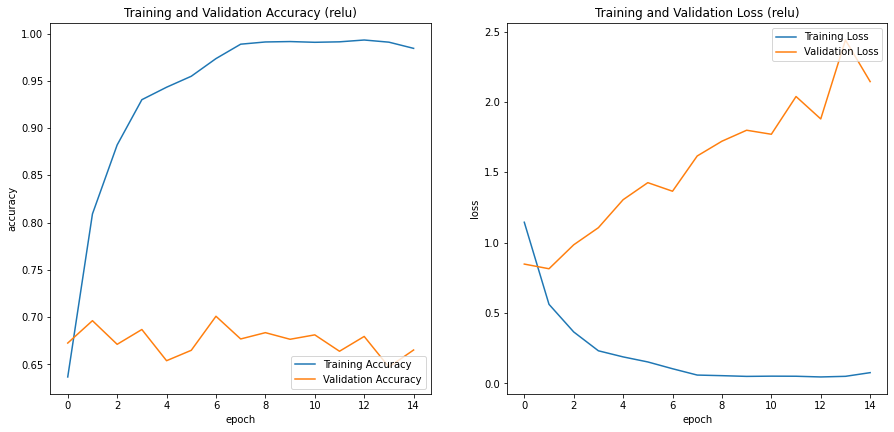
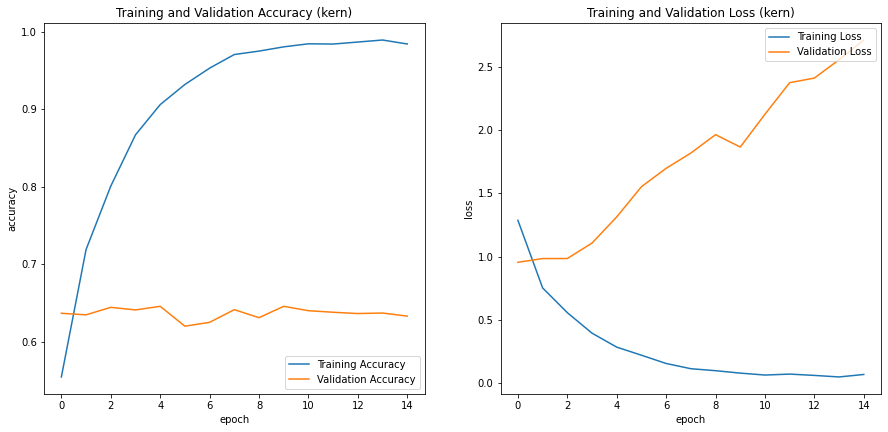


Overall, Relu performed much better than Sigmoid with a validation accuracy of 66% vs 47%. Neither model showed an increase in validation accuracy with an increasing number of epochs. Both models overfit the training data, especially the Relu model, which showed a 33% difference in training vs validation accuracy.

Relu appears to be a superior activation function for this application. It will be used in all following tests.

### 3x3 Kernel vs 5x5 Kernel

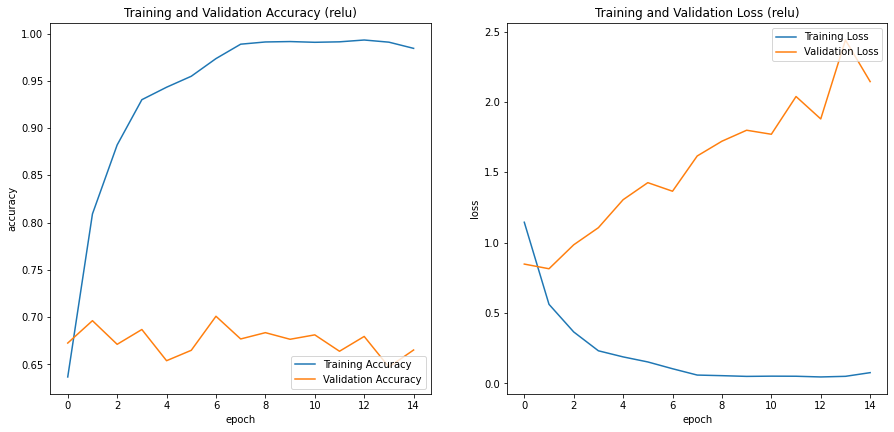
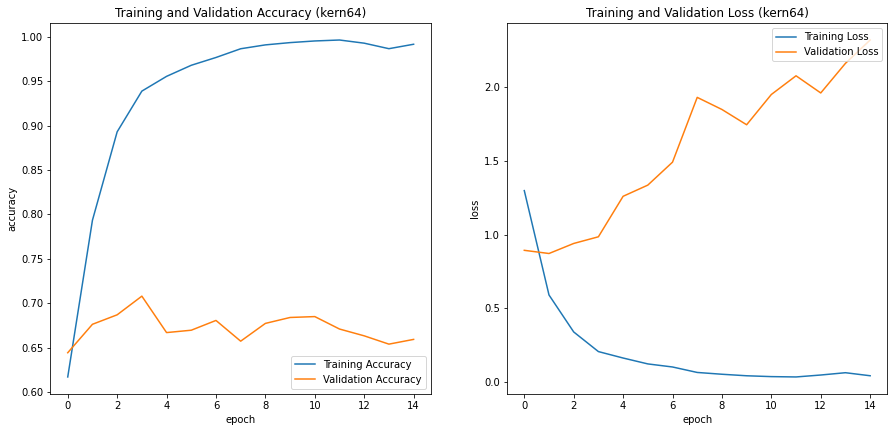
|  |  |  |
| --- | --- | --- |
| **Kernel Size** | 3x3 | 5x5 |
| Runtime per epoch | 58s | 71s |
| Training accuracy | ~99% | ~99% |
| Validation accuracy | ~66% | ~64% |

Using a larger kernel in the convolutional layer increases runtime by about 13s (22%) but has no significant effect on accuracy or overfitting.

### 32 Filters vs 64 Filters

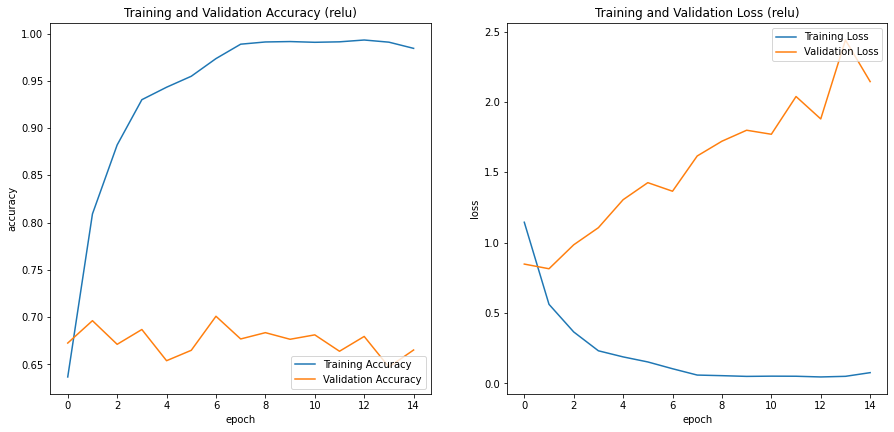
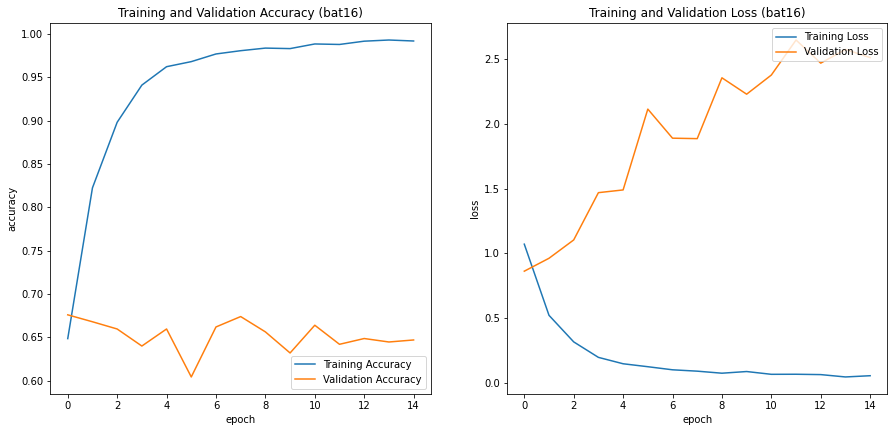
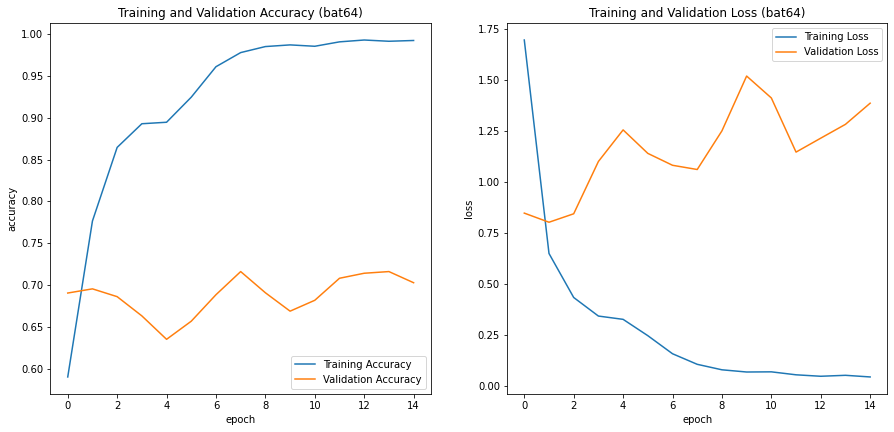
|  |  |  |
| --- | --- | --- |
| **Filter Count** | 32 | 64 |
| Runtime per epoch | 58s | 101s |
| Training accuracy | ~99% | ~99% |
| Validation accuracy | ~66% | ~66% |

Doubling the filter count in the convolutional layer increases runtime by about 43s (74%) but has no significant effect on accuracy or overfitting. It may be worth testing a larger filter count with increased max pooling parameters afterward to reduce runtime.

### 16 vs 32 vs 64 Batch Sizes

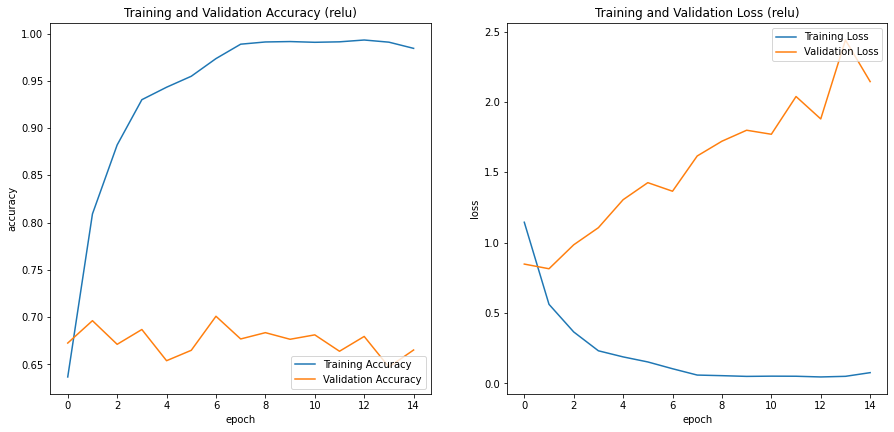
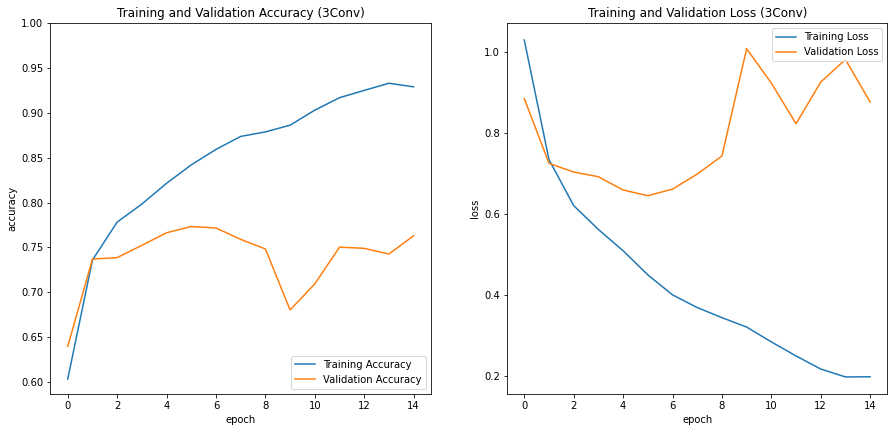
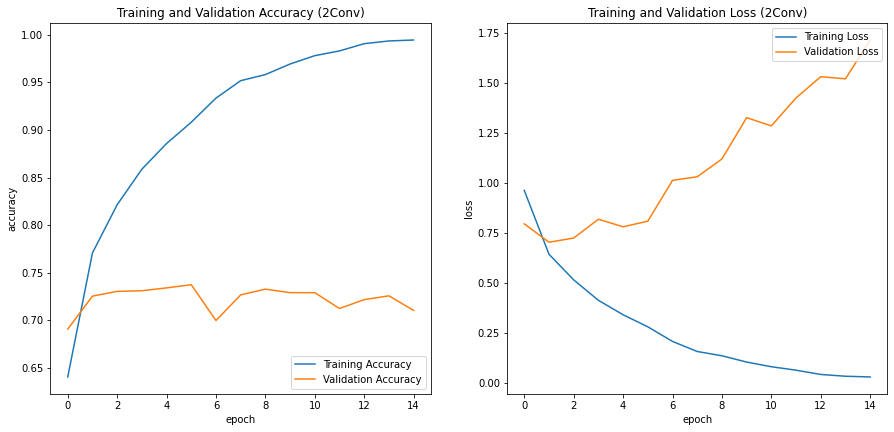
|  |  |  |  |
| --- | --- | --- | --- |
| **Batch Size** | 16 | 32 | 64 |
| Runtime per epoch | 61s | 58s | 56s |
| Training accuracy | ~99% | ~99% | ~99% |
| Validation accuracy | ~65% | ~66% | ~70% |

Increasing the batch size slightly improved both runtime and validation accuracy. A batch size of 64 will be used moving forward.

### Number of Convolutional Layers

|  |  |  |  |
| --- | --- | --- | --- |
| **# of Conv Layers** | 1 | 2 | 3 |
| Runtime per epoch | 58s | 92s | 101s |
| Training accuracy | ~99% | ~99% | ~93% |
| Validation accuracy | ~66% | ~72% | ~75% |

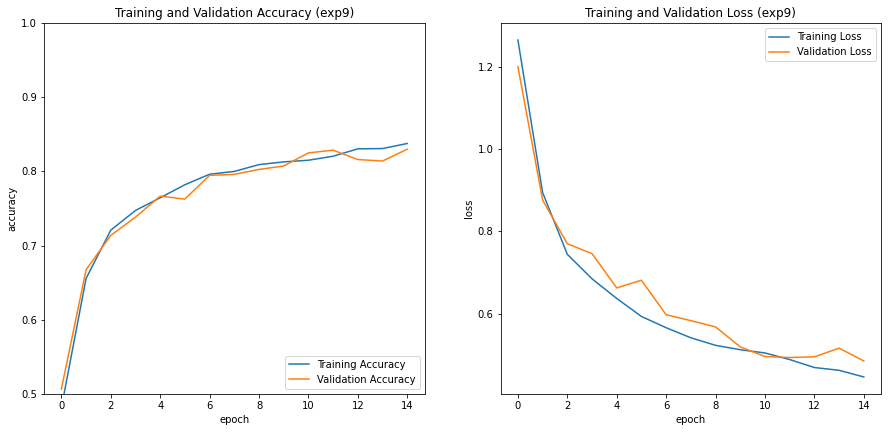
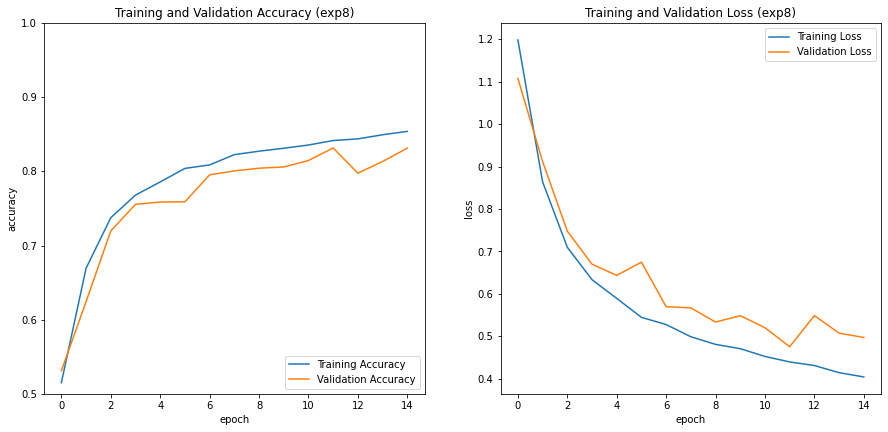
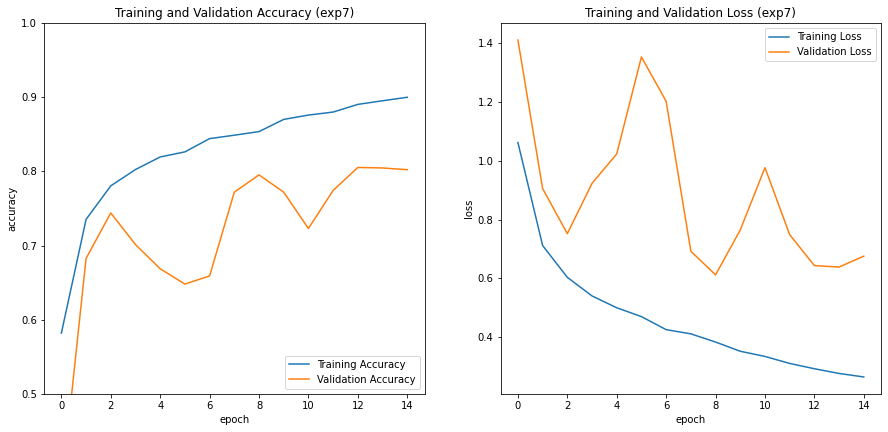
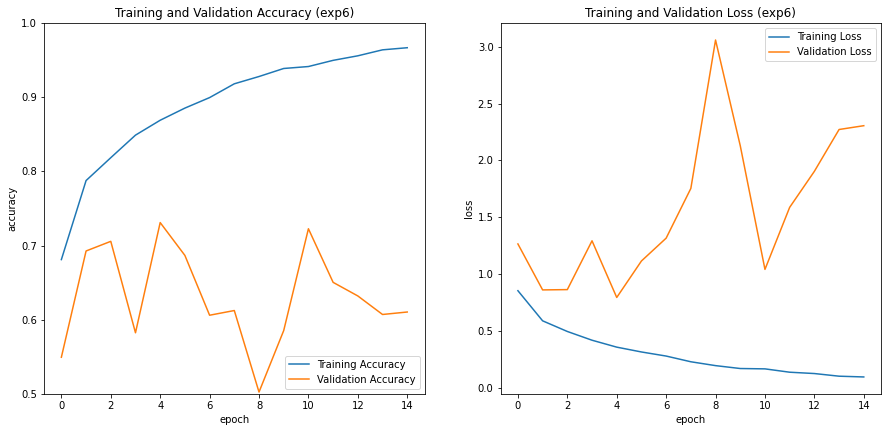
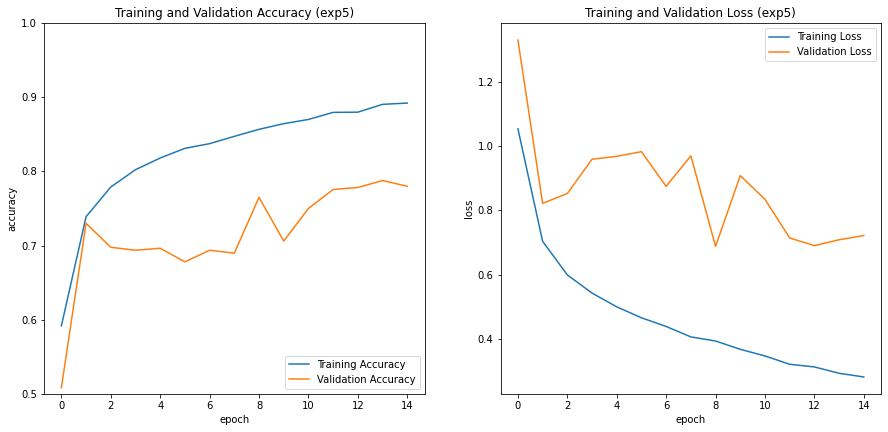
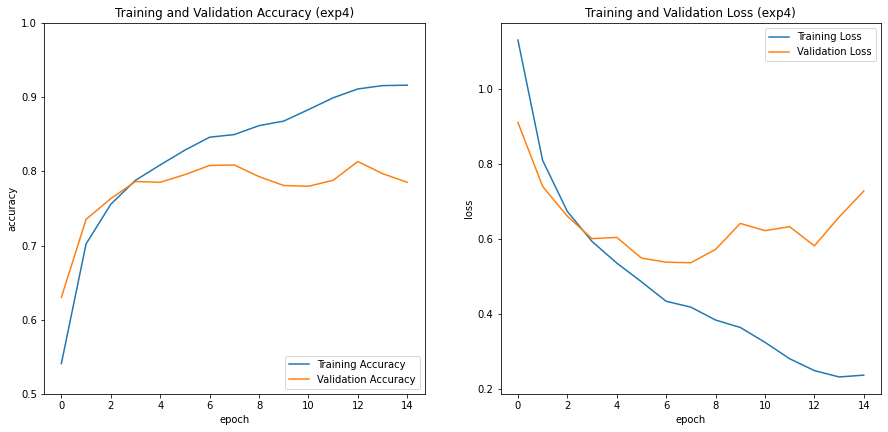
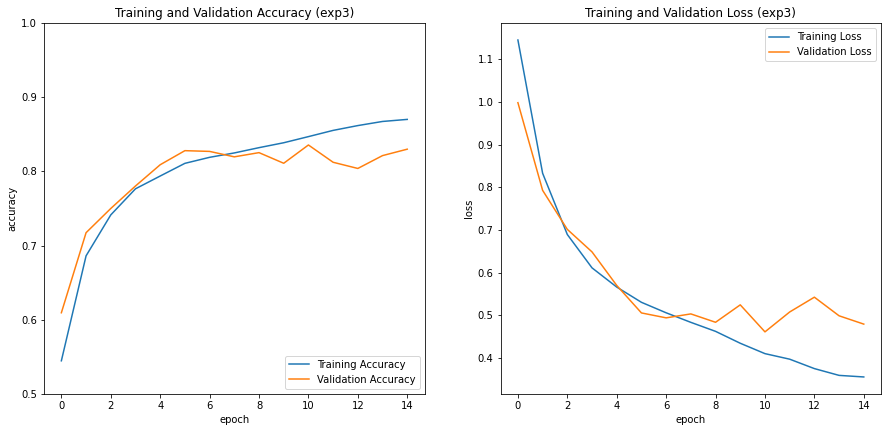
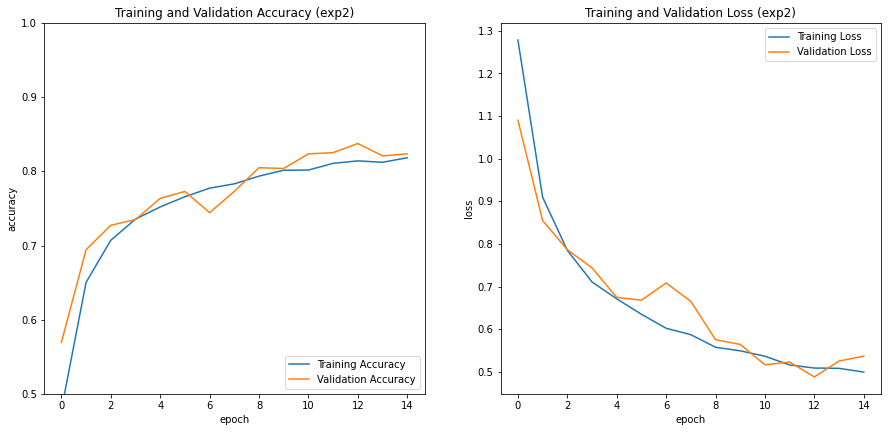
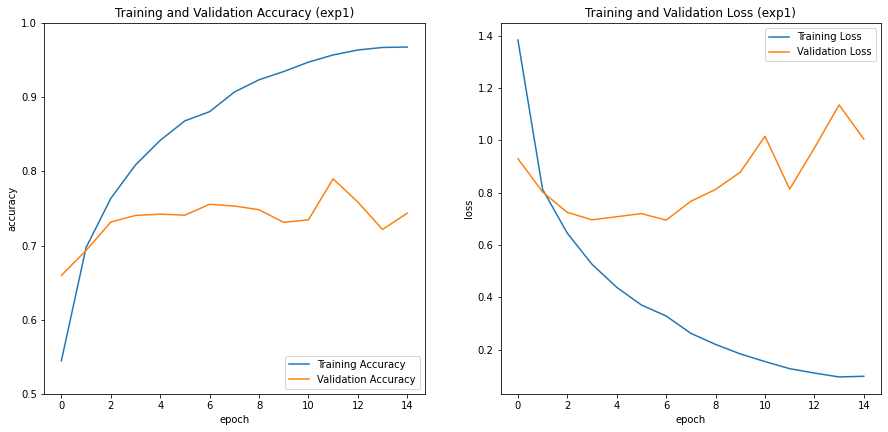
Adding convolutional layers increased validation accuracy at the expense of runtime. At three convolutional layers, overfitting was reduced with the difference between training and validation accuracy dropping to 18%.

## Refining Hyperparameters

### Experimental Models

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Parameter Summary | Stats | Result Summary |
| Exp1 | Two convolutional layers with max pooling after each to reduce runtime and dropouts after each to reduce overfitting. An additional fully connected layer is added between the flatten and classification. | Time/epoch: 140s  Training acc: ~96%  Validat acc: ~74% | Model is not efficient enough (takes too long to run). It is slower, less accurate, and overfits more than the 3conv model above. |
| Exp2 | Three convolutional layers with max pooling after each. Max pooling size is increased for the first two layers to improve runtime. Dropouts are added after each max pool to reduce overfitting. Two additional fully connected layers are added between the flatten and classification. | Time/epoch: 60s  Training acc: ~83%  Validat acc: ~81% | Increased max pooling parameters greatly decreases model runtime. Adding additional fully connected layers and convolutional layers helps improve accuracy. |
| Exp3-Exp9 | Same as Expr2. Each model experiments with different dropouts, filter counts, and/or batch normalization | Batch normalization decreased accuracy and increased overfitting, so it was not used in the final model. Increased filter counts within the convolutional layers coupled with max pooling had no significant effect on runtime but also showed no increase in overall model accuracy and no decrease in overfitting. | |

Model accuracy:



## Final Model

Rescaling: 0-255 RGB values to 0-1

Model: "sequential\_19"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

rescaling\_19 (Rescaling) (None, 150, 150, 3) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_41 (Conv2D) (None, 150, 150, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_41 (MaxPooling (None, 37, 37, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_25 (Dropout) (None, 37, 37, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_42 (Conv2D) (None, 35, 35, 32) 9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_42 (MaxPooling (None, 11, 11, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_26 (Dropout) (None, 11, 11, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_43 (Conv2D) (None, 9, 9, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_43 (MaxPooling (None, 4, 4, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_27 (Dropout) (None, 4, 4, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_19 (Flatten) (None, 1024) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_38 (Dense) (None, 256) 262400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_39 (Dense) (None, 128) 32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_40 (Dense) (None, 6) 774

=================================================================

Total params: 324,710

Trainable params: 324,710

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

------------------------------------------------------------

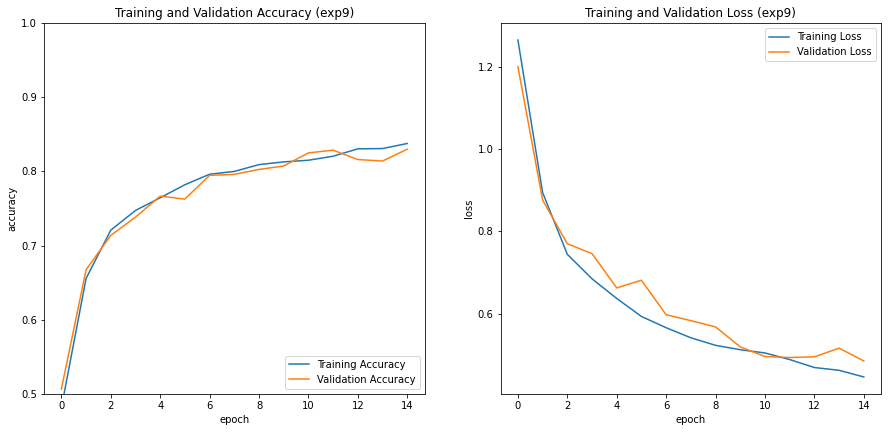
Conv: 32 filters, 3x3 kernel, 1 stride, relu  
MaxPool: 4x4  
Dropout: 0.4 rate

Conv: 32 filters, 3x3 kernel, 1 stride, relu  
MaxPool: 3x3  
Dropout: 0.4 rate

Conv: 64 filters, 3x3 kernel, 1 stride, relu  
MaxPool: 2x2  
Dropout: 0.4 rate

------------------------------------------------------------

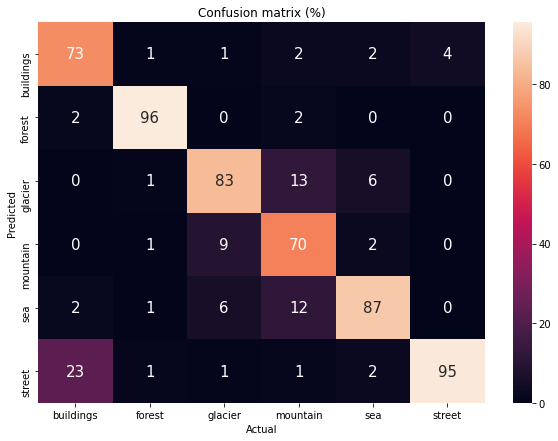
Flatten  
Dense: 256 nodes, relu  
Dense: 128 nodes, relu  
Dense (classification layer): 6 nodes, softmax



### Summary

The final model shows a validation accuracy of about 83% with no apparent overfitting at 15 epochs and has a runtime of 61s per epoch: comparable to the baseline model. Notable weaknesses in the model:

* 23% of buildings incorrectly classified as streets
* 13% of mountains incorrectly classified as glaciers & 12% incorrectly classified as sea
* 9% of glaciers incorrectly classified as mountains & 6% incorrectly classified as sea

Overall accuracy per class:

* Buildings: 73%
* Forest: 96%
* Glacier: 83%
* Mountain: 70%
* Sea: 87%
* Street: 95%

### Further Prediction Visualization

The final model has been saved and submitted with this writeup and Jupyter notebook. If you’d like, download the seg\_pred image folder from <https://www.kaggle.com/puneet6060/intel-image-classification> and place it in the root directory of this notebook. At the bottom of the notebook is a script that will test random images in the folder and show a breakdown of classification probabilities for each picture.

Example:

