Classifying Members of *Ursidae* with Neural Networks

Shaney Flores

Spring 2021

Abstract

Rationale. The taxonomic family *Ursidae* consists of bears that include pandas, grizzlies, and black bears. Each bear species has unique morphological features distinguishing it from other members of *Ursidae*, such as pointed or rounded ears and snouts. Can we train a convolutional neural network to help distinguish different members of the family *Ursidae*?

Methods. Over 1000 digital colored images of panda, grizzly, and black bears were downloaded from an Internet image search. All images were resized to 256x256, normalized, and fed into several neural network architectures. Different network architectures were compared, and regularization techniques were applied to improve network performance. The best-fitting architecture was then trained and tested on augmented data to determine if the model could generalize to novel data. We then evaluated three pre-trained networks (VGG16, ResNet50, and DenseNet121) to determine whether they could perform better than our best performing network.

Results. The best-fitting network was able to achieve 89.71% accuracy on the test data. Increasing the number of layers or filters did not improve accuracy while decreasing layers or filters resulted in worse performance. When evaluated on augmented data, our best-fitting model still accurately classified bears at a comparable level. Both batch normalization and dropout helped the best-fit model when individually applied, but combining them did not help the model. The VGG16 performed as well as the regularized best-fit model, but the ResNet50 performed worse than any model. The DenseNet121 achieved over 99% accuracy but had variable performance on the validation data.

Conclusions. We were able to develop a best-fit model that could correctly classify bears over 89% of the time on non-augmented data and at nearly similar levels on augmented data. Regularization improved performance, increasing it to 92%. Pre-trained models had mixed success with the DenseNet121 outperforming all models. Further studies on the DenseNet121 are needed to verify the result is valid.

Contents

List of Tables

List of Figures

1 Introduction

Bears are members of the family *Ursidae* that includes panda, grizzly, and polar bears, with each species having morphological features distinguishing it from the others. For example, polar bears are leaner and taller than grizzlies or pandas while pandas are shorter and rounder than either grizzly or polar bears. Can a convolutional neural network help distinguish the different species of bear using images downloaded from a Bing image search?

2 Methods

2.1 Data Description

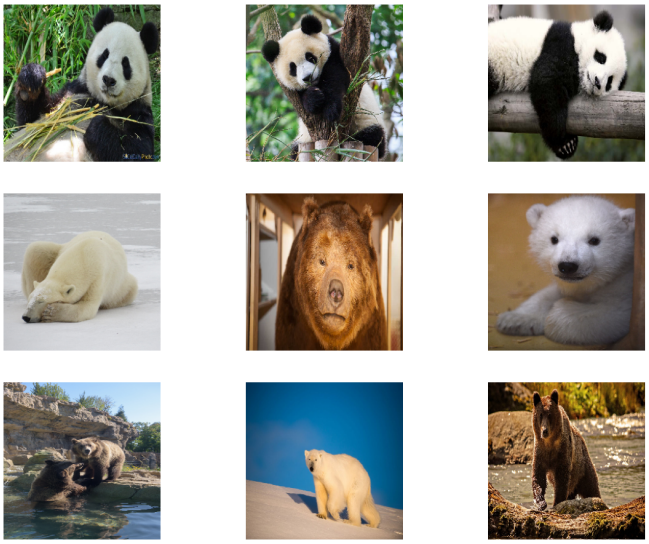

Figure 1: Representative images of classes

Colored images for polar, panda, and grizzly bears were batch downloaded from a Bing image search and curated to remove other animals or artistic renditions of the bear, such as paintings or drawings (representative images are shown in Figure 1). The final data set includes 1,213 images of polar, panda, and grizzly bears (see Table 1 for breakdown of image set).

Class	Frequency
Grizzly Bear	398
Panda Bear	403
Polar Bear	412

Table 1: Counts for each class

2.2 Data Normalization and Splitting

Images were reshaped to 256 x 256 scaling with pixel values divided by 255 to normalize values to a scale of [0,1]. All images were kept in their native RGB color scale. For model fitting, the data

set was randomly split into three groups: training set, validation set, and testing set. The validation and testing sets each consisted of 20% of the data. The remaining 60% were assigned to training.

One portion of our experiments required the construction of an overfitted model using all available data. For this, we created copies of the image set and randomly divided it into two groups: overfit training and overfit validation. Although it is advisable to randomly sort any data set before every attempt at overfitting to improve model reliability, the model's reproducibility takes a significant hit. To ensure reproducible results, no further random assignment was performed on the overfit data set after the initial splitting.

3 Modeling and Results

3.1 Model Overfitting

To test whether a model would be sufficiently powered to generalize to novel data, three overfitted networks were created using the overfit data set. A baseline overfitted network was first created to provide a comparison (Model 1). Two additional networks were created that either increased the number of layers (Model 2) or increased the number of filters in each layer (Model 3). Each model relied on an Adam optimizer with the default learning rate. All layers, except the last, utilized a relu activation function. Model 1 was trained for 100 epochs with a batch size of 64; however, after noticing the network reach maximum performance many epochs before reaching 100, Models 2 and 3 were trained for only 50 epochs. Performance and descriptions of all three models are provided in Figure 2.

Model 1:
- *3 convolutional layers*: 64, 32, and 16 filters; 5x5 kernel; relu
- *1MaxPool*: 5x5 filter
- *4 dense layers*: 40, 20, 10, and 3 filters; all relu except for last layer

Model 2:
- *7 convolutional layers*: 64, 32 (x5), and 16 filters; 3x3 filter; relu
- *1 MaxPool*: 5x5 kernel
- *5 dense layers*: 80, 40, 20, 10, and 3 filters; all relu except for last layer

Model 3:
- *3 convolutional layers*: 128, 64, and 32 filters; 3x3 kernels; relu
- *1 MaxPool*: 5x5 kernel
- *4 dense layers*: 120, 60, 30, and 3; all relu except for last layer

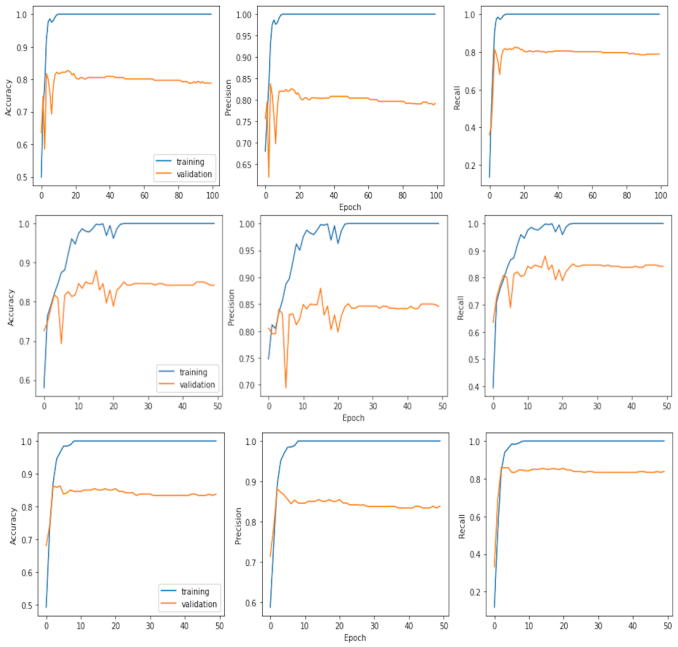

Figure 2: Learning curves for overfit models

Each overfitted network was able to correctly identify training images of bears by around the 10th epoch. Neither increasing the number of layers or the number of filters substantially improved

performance for any outcome measure. Validation performance reached a plateau around the 20th epoch for Model 1, approximately the 25th epoch for Model 2, and approximately the 3rd epoch for Model 3. Interestingly, increasing the number of filters per layer resulted in more stable training and higher validation accuracy while increasing the number of layers actually resulted in more erratic learning until about the 25th epoch.

As the networks were able to quickly overfit the data, it is possible they may be relying heavily on either background features of the images to aid in learning (e.g., polar bears are more likely to be in wintry environments than grizzly or panda bears) or on color.

3.2 Model Fitting

We next attempted to create a model that could actually fit the data. To prevent over-training, an early stop callback was created that ended training after 15 consecutive epochs where the minimum validation loss did not change significantly across epochs.

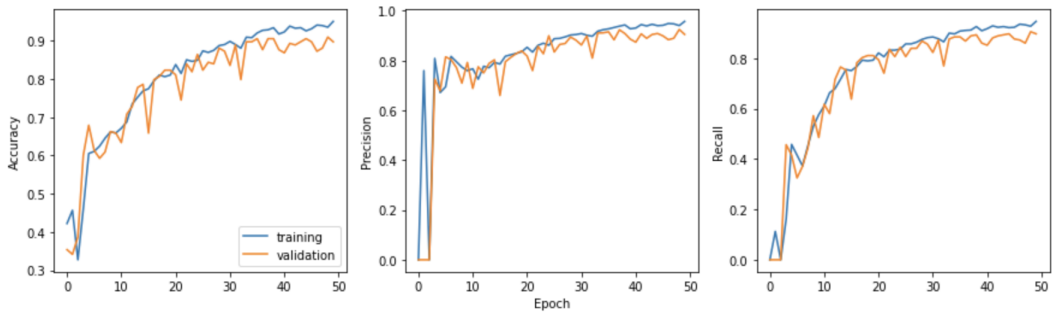

Figure 3: Learning curves for best-fit model

We found the best network consisted of three convolutional layers containing 16, 8, and 4 filters with a 3x3 kernel for each layer. Between each of these convolutional layers was a MaxPool operation with a 5x5 filter. After flattening the input, data was fed it into two dense layers with 5 and 3 filters, respectively. All layers, except the final layer, used a relu activation function. This model was trained for 50 epochs with 64 samples taken per training session. Performance plots for this network can be seen in Figure 3.

Model	Accuracy	Loss	Precision	Recall
Best fit	89.71%	0.33	0.90	0.89
Decrease layers	85.19%	0.45	0.86	0.84
Increase layers	89.17%	0.42	0.90	0.90
Decrease filters	60.08%	0.78	0.83	0.29
Increase filters	89.71%	0.41	0.90	0.89

Table 2: Performance of each fitted model evaluated on test data

Learning for this network was fairly normal as both training and validation followed in step with one another and performance improved across epochs. It should be noted that the model did display some erratic performance during validation. After training, this model was evaluated on the testing set where it achieved an accuracy rate of 89.71%, indicating an optimal model for this type of problem. However, as stated before, color or background environment may be playing a significant role

in this accuracy rate and should be evaluated.

Using our best-fitting model as a baseline, we compared several other architectures that either increased or decreased the number of layer or the number of filters per layer. Results from training of these models are presented in Table 2. All of these models were trained for 50 epochs with a batch size of 64 as in the best fit model.

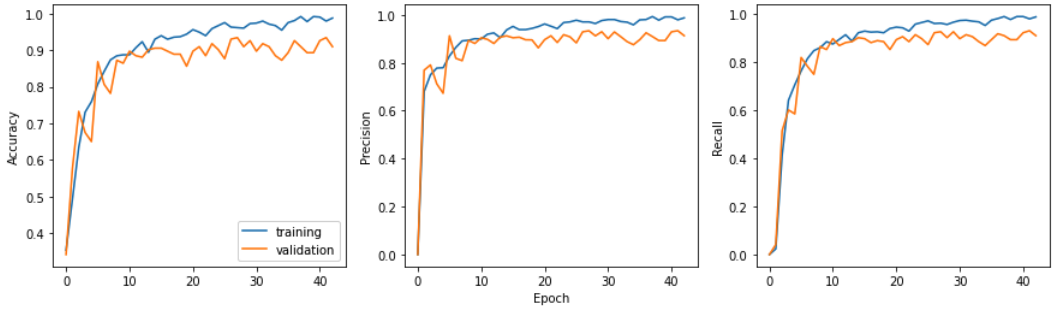

Figure 4: Learning curves for best-fit model with additional dense layer

In general, increasing the number of layers or number of filters per layer appeared to produce models that were equivalent in performance to the best-fit model (see Figures 4 and 5 for the learning curves of these two models). For the model that increased the number of layers, a single dense layer consisting of 10 filters was added to the best-fit model. For the model that increased the number of filters per layer, the number of filters in every layer of the best-fit model was increased by a factor of two. Conversely, decreasing either the number of layers or filters produced worse performing models. This suggest our current best-fit model may be the smallest architectures for correctly classifying bears.

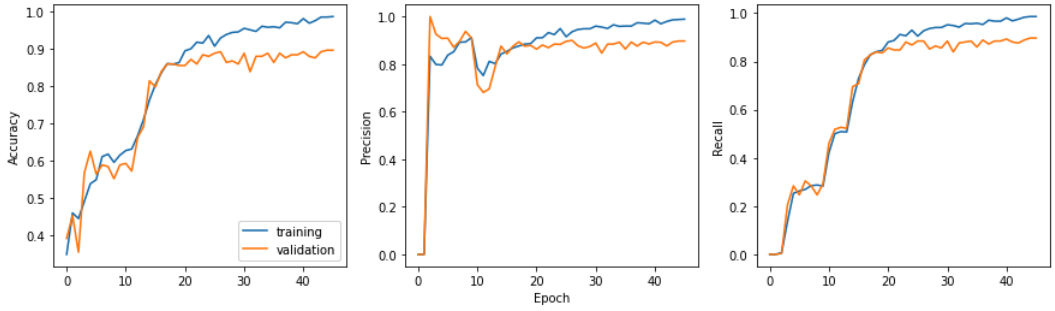

Figure 5: Learning curves for best-fit model with x2 more filters

3.3 Evaluating Best-Fit Network using Augmented Data

To ascertain whether our best-fit network can handle a variety of new types of data, we next trained it on augmented data and evaluated its performance. Each augmentation was additive to determine whether a particular augmentation harmed model accuracy.

Model	Accuracy	Loss	Precision	Recall
Augmentation 1	90.12%	0.31	0.91	0.88
Augmentation 2	89.71%	0.27	0.90	0.90
Augmentation 3	87.65%	0.34	0.89	0.86
Augmentation 4	84.77%	0.43	0.87	0.82

Table 3: Performance of best-fit model for each augmentation on test data

For Augmentation 1, images were simply shifted vertically and horizontally by 20%, as well as randomly flipped across the horizontal axis. For Augmentation 2, we added a 20% zoom on the images. For Augmentation 3, we added a rotation to each image of 10°. Finally, for Augmentation 4, the number of color channels was reduced from 3 to 1 to test whether color was significant feature for model performance.

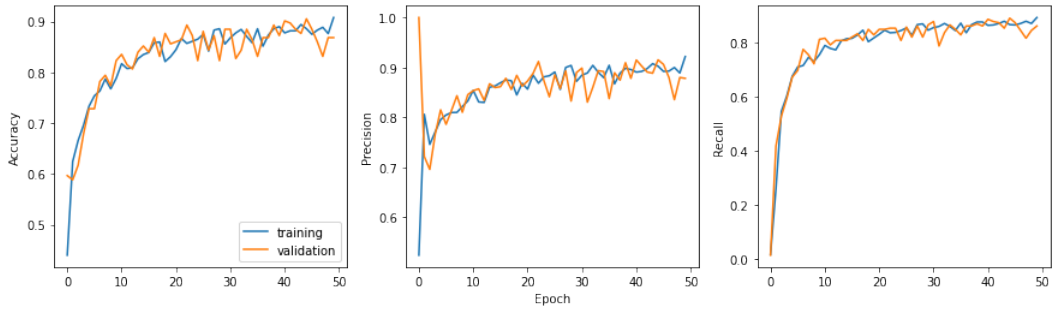

Figure 6: Learning curves for best-fit network trained with Augmentation 3

Accuracy, loss, precision and recall for each Augmentation is shown in Table 3. As expected, simple shifts, flips, zooms, and small rotations did not significantly impact network performance. Color appeared to produce the biggest difference as accuracy fell to 84.77% from a baseline of 89.71% for the best-fit model trained with non-augmented data, suggesting color may be an important but not critical feature for the network.

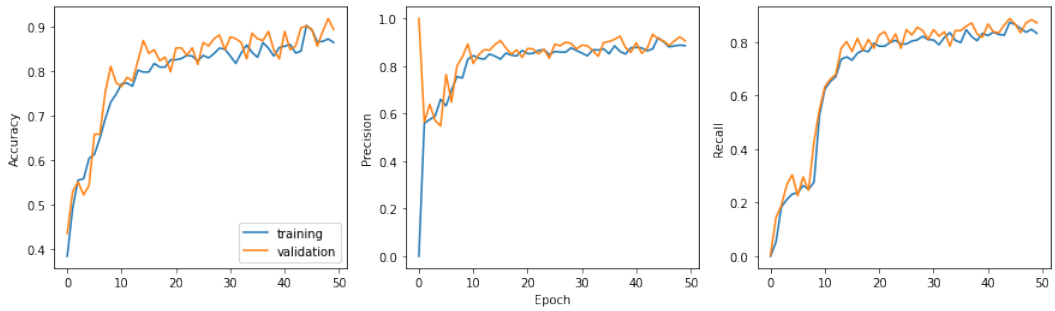

Figure 7: Learning curves for best-fit network trained with Augmentation 4

Performance plots for Augmentations 3 and 4 are shown in Figures 6 and 7, respectively. Augmentation 3 was able to learn faster than the best-fit model but reached near equivalent performance by end of training. Augmentation 4 was also able to learn at a reasonable pace, reaching a near plateau

of performance by the 15th epoch. However, it's overall accuracy was lower than the best-fit model. Our model does appear to be capturing features of the images that are not color-dependent.

3.4 Applying Model Regularization

One way to improve predictive accuracy is adding regularization. To assess whether regularization methods improved model accuracy, we added batch normalization, dropout, and L2 regularization to our best-fit model. In contrast to previous experiments, we analyzed the impact of each regularization procedure independent of the others by adding only batch normalization, dropout, or L2 regularization to the best-fit model. This was done after we discovered combining both batch normalization and dropout performed no better than our best-fit model (further described below).

Regularization Method	Accuracy	Loss	Precision	Recall
Batch Normalization	92.59%	0.21	0.93	0.93
Dropout	91.36%	0.39	0.92	0.88
L2 Regularization	86.42%	0.48	0.88	0.85
Batch Normalization + Dropout	84.77%	0.52	0.88	0.78

Table 4: Performance of regularized best-fit model on test data

Performance metrics for each regularization method are shown in Table 4. Adding batch normalization or dropout layers modestly helped overall model performance compared to the non-regularized best-fit model. For these models, batch normalization was added before each convolutional layer and dropout was added after each convolutional layer. Of the two methods, batch normalization granted the biggest increase in performance. Learning curves for batch normalization are shown in Figure 8.

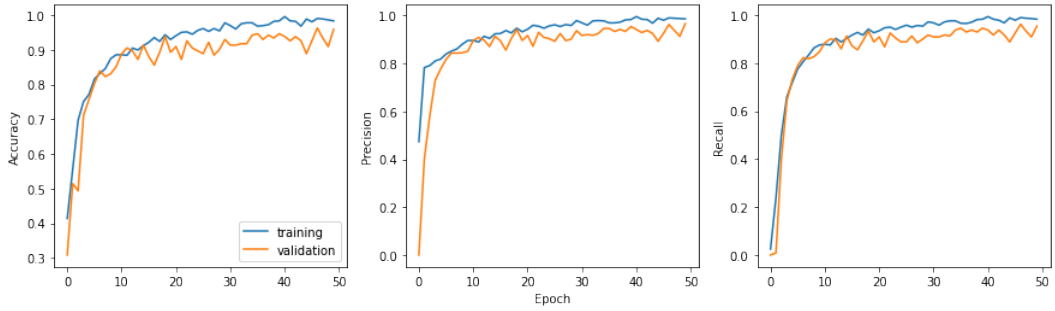

Figure 8: Learning curves for best-fit network with batch normalization

Training for the model using batch normalization followed a relatively stable trend and achieved near 95% accuracy, precision, and recall in the final epoch. Validation accuracy was still erratic but increased in tandem with training accuracy, indicating batch normalization was helpful to some extent for the network.

As previously mentioned, we attempted to combine regularization techniques to see whether this would benefit the best-fit model. For this, a batch normalization was added before and a dropout layer was added after each convolutional and maxpool operation. Learning curves for this regularization scheme are shown in Figure 9. Unfortunately, a model utilizing both batch normalization

and dropout in this manner performed no better than the best-fit model, achieving an accuracy of 84.77% for the test data. It is possible that a more conservative application of batch normalization and dropout layers could be helpful (e.g., limiting the number of dropouts in the best-fit model to only one or two, potentially after the last convolutional and maxpool operations).

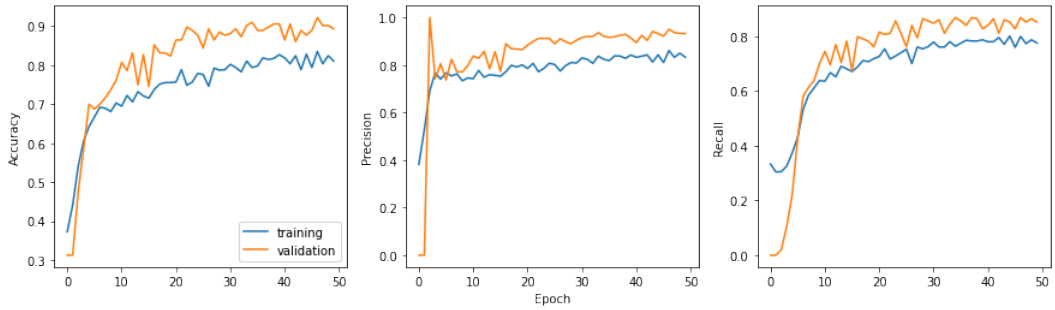

Figure 9: Learning curves for best-fit network with batch normalization and dropout

Interestingly, adding L2 regularization to each convolutional layer decreased accuracy relative to the best-fit model, although not substantially. Learning curves for the best-fit model with L2 regularization are shown in Figure 10. Training was relatively stable for L2 regularization; however, it only achieve about 90% accuracy during training as opposed to the 95% for batch normalization. These results suggest that a more conservative application of batch normalization may actually help our model.

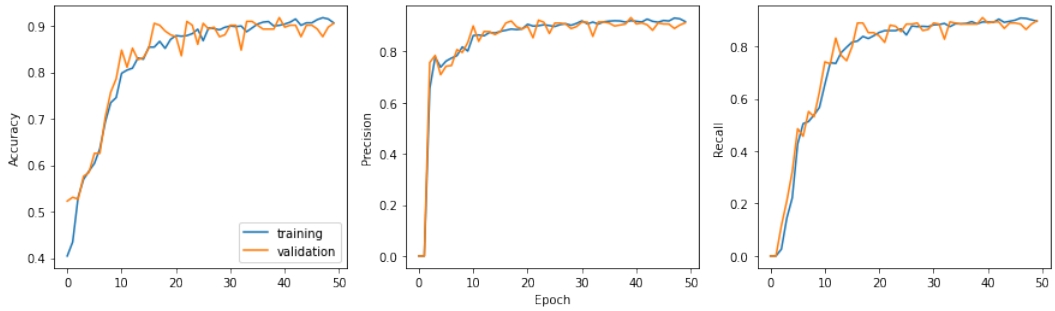

Figure 10: Learning curves for best-fit network with L2 regularization

3.5 Applying Pre-trained and Recent Network Architectures

Our best-fit model so far is achieving 89% accuracy on its own and close to 92% accuracy with batch normalization on our test data set. But the deep learning field is ripe with pre-trained models for image classification that could be re-purposed for the problem at hand. Would one of these pre-trained models actually perform better than our current optimal architecture? We investigated this possibility by attempting to fine-tune three existing network architectures: the early generation VGG16, the mid-generation ResNet50 and more recent DenseNet121.

For fine-tuning, the weights of the last convolutional layer are adjusted through training while the remaining weights stay constant. All networks were trained for 50 epochs using an rmsprop optimizer

Network	Accuracy	Loss	Precision	Recall
Best-Fit	89.71%	0.33	0.90	0.89
Best-Fit + Batch Normalization	92.59%	0.21	0.93	0.93
VGG16	93.00%	0.35	0.93	0.93
DenseNet121	99.18%	0.06	0.99	0.99
ResNet50	77.37%	3.36	0.77	0.77

Table 5: Performance of regularized best-fit model on test data

with the default learning rate. The early stop callback from our evaluations was used to ensure no model overfitted the data. Performance of these models on the test data, as well as our best-fit and highest performing regularized best-fit model, are shown in Table 5.

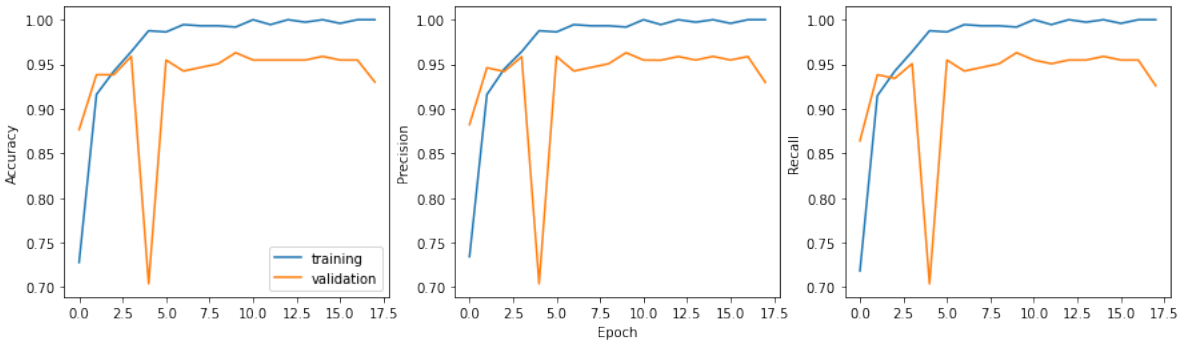

Figure 11: Learning curves for VGG16 network

The VGG16 network performed only slightly better than our regularized best-fit model, further suggesting our regularized best-fit model is more efficient than the layer-heavy VGG16. The ResNet50 surprisingly performed worse than our best-fit model. However, the recent DenseNet121 network achieved over 99% accuracy on classifying bears! Learning curves for both the VGG16, DenseNet121, and ResNet50 models are shown in Figures 11, 12, and 13, respectively.

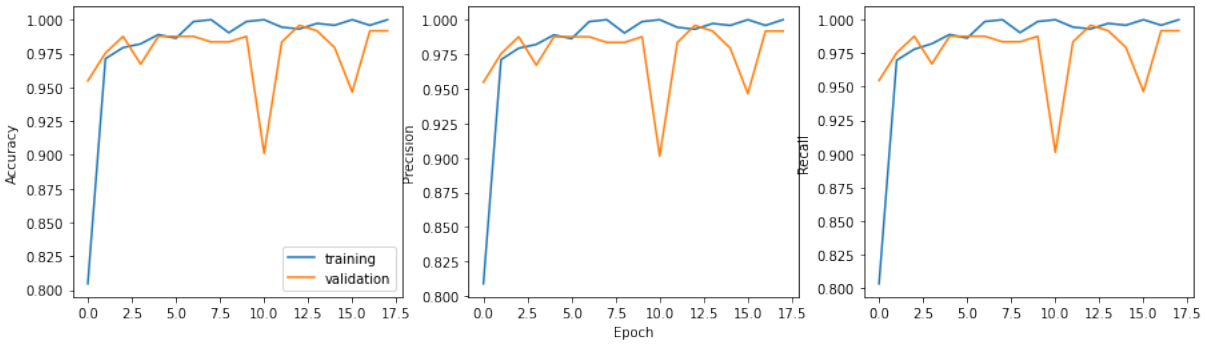

Figure 12: Learning curves for DenseNet121 network

The VGG16 and DenseNet121 networks both ended training by the 18th epoch while the ResNet50 ended by the 24th. Both VGG16 and DenseNet121 achieved peak performance at different rates: the

VGG16 took five epochs to reach near perfect accuracy on training, but the DenseNet121 achieved the same level by the 2nd epoch. Training was much more gradual for ResNet50, reaching maximal performance by around the 20th epoch. However, unlike the other two, ResNet50 appeared to overfit the data. It is possible opening more convolutional layers of the ResNet50 for training could improve accuracy.

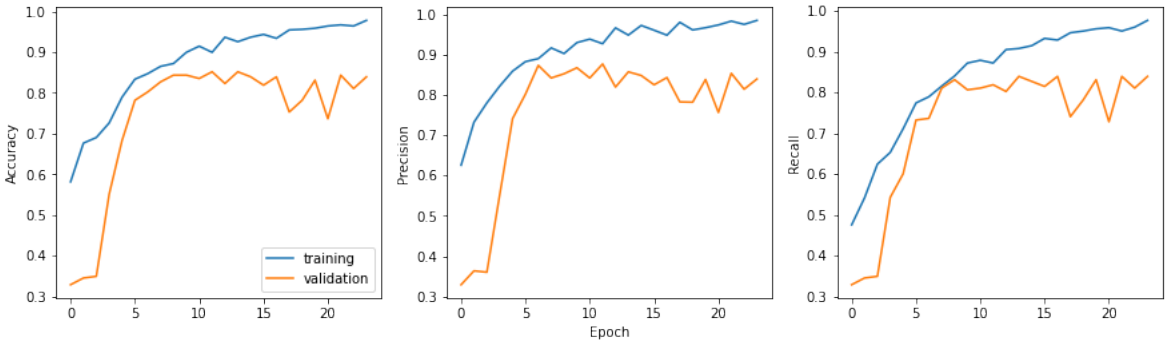

Figure 13: Learning curves for ResNet50 network

Pre-trained networks do appear to learn and classify bears better than our manually generated network in some cases. This may be due to the fact that the DenseNet121 is a newer generation architecture compared to the VGG16 or ResNet50 and has had the benefit of refined training and development that would suit this type of problem.

4 Conclusions

Our goal was to design and evaluate networks that could accurately classify bear species based on images downloaded from an Internet search site. To this end, we tested several manually generated models and three pre-trained networks. We further evaluated our manually generated models on augmented data to ensure that the best-performing model could generalize to novel data.

We found the best network included three convolutional layers with 3x3 kernels containing 16, 8, and 4 filters, respectively, and a 5x5 MaxPool operation between each convolution. The output of these convolutional layers was then fed into a small dense network of 5 and 3 filters each. Without regularization, this network performed fairly well on the task, achieving 89.71% classification accuracy. Increasing the number of layers or filters per layer did not seem to help model performance, suggesting our network is perhaps one of the smallest base architectures that could be built to accurately classify bears. This was further supported when decreasing the number of layers or filters significantly hurt accuracy.

When the best fit model was exposed to augmented data, it performed decently well, achieving comparable accuracy to non-augmented data. This gives us confidence the model is likely detecting unique features to every bear and using those to help classify them. We were given further evidence of this as when color is removed from our images accuracy fell modestly from 89% to 84.77%. Additionally, in all of our augmented data experiments, the augmentations were additive, and we observed the expected decrease in accuracy as more augmentations were added. This decrease is attributable to the model having a harder time using the more discriminating or informative features to make an accurate classification the more those features are manipulated or blended together.

Model performance increased modestly to 92.59% when batch normalization added before each convolutional layer. Although adding dropout after each convolution also marginally increased performance to 91.36%, combining both regularization techniques actually damaged performance, reducing it to 86.42%. It is possible that combining these two regularization techniques may be helping the model too much so that it begins to underfit the data. However, our approach was also very aggressive in combining batch normalization and dropout around each convolution operation. It is entirely likely that a more conservative application of applying dropout and batch normalization may actually help performance but to what extent is unknown.

Finally, pre-trained models had mixed success with performing the classification task relative to our best-fit model. While the older generation VGG16 architecture did perform better than our best-fit model, its performance was equivalent to the less complex regularized best-fit. The mid-generation ResNet50 actually performed worse than any of our evaluated models, but this is possibly due to having only one layer of the ResNet50 available to learn during training. Increasing the number of convoluational layers available to learn may help the ResNet50 perform better, but the ResNet50 was already begin to overfit.

But most striking was that the newer generation DenseNet121 achieved an outstanding 99.18% accuracy on the classification task–most likely due to the DenseNet121 having a more modern architecture carefully calibrated to this sort of task compared to its predecessors. But this comes with a strong caveat. We would prefer a model that learns more gradually over the course of many epochs and is consistent with its accuracy during both training and validation, but the DenseNet121 had quite an erratic learning curve during validation. Further tests with the DenseNet121 are needed to verify the high accuracy is a true result and not a statistical fluke.

Overall, models can be designed manually and from pre-trained networks that classify images of bears to an acceptable level for the task at hand.