Impact of transfer learning techniques in identifying the type of damage on an image for car insurance claims

Ronald Nhondova
Master in Interdisciplinary Data Science (MIDS)
Duke University
ronald.nhondova@duke.edu

Abstract

The main objective of the project is to find the best candidate model to automate the classification of the location of damage on the car in a supplied image. To achieve this, 3 transfer learning techniques are considered were pre-trained models are used as (1) feature extractors, (2) for weight initialization and model fine-tuning then carried out and lastly (3) combination of feature extractor and weight initialization. As part of applying transfer learning to the task, comparison of different optimizers and learning rates is carried out to identify sensitivities of the above techniques. The results identify transfer leaning techniques (2) and (3) to be more sensitive to optimizer and learning rate. The best candidate model for the task, selected based on accuracy, is a wide residual network.

1 Background

Claims processing for car insurance can be an administratively costly process as the time investment required from claim adjusters can be intensive. Most expenses incurred by insurance companies are passed on to policyholders, so any process optimizations should be beneficial to both the company and the policyholder.

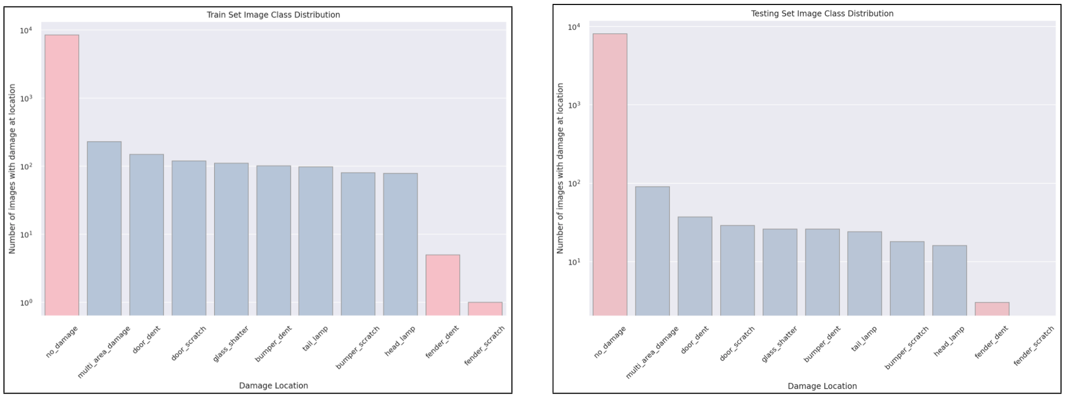
Previous work with respect to identifying location of damage on cars has been done for example by Jayawardena[1], with a focus on applying different novel techniques in order accomplish their objectives. Thanks to the efforts of those authors, the viability of the idea has been shown to be feasible.

In many of the previous approaches, the authors would train models from scratch[2] but this can present multiple challenges especially in cases when insufficient data is available. In addition, these approaches might fail to leverage knowledge learned from a large and diverse source task like for example ImageNet.

The main objective of the project is to find the best candidate model to automate the classification of the location of damage on the car in a supplied image. To achieve this, 3 transfer learning techniques are considered as follows:

- Fixed Features – were pre-trained model is used as feature extractor and only train last layer.
- Finetune – were pre-trained model weights are used for initialization and model finetuning is then carried out.
- Hybrid – is mixture of the above two methods, with first half (fifty percent) of layers in pre-trained model having fixed weights and finetuning the remaining.

ECE 590-03 (2020 Fall) final report.



(a) Training set (b) Test set

Figure 1: Image distribution by label for training and test set.

(a) Head lamp (b) Glass shatter (c) Door dent (d) Multi area (e) No damage

Figure 2: Sample images being used for classification.

In addition, the effect of using either Stochastic Gradient Descent (SGD) or Adaptive moment estimation (ADAM) as optimizers, with various learning rates is observed for the above transfer learning techniques.

2 Method

2.1 Data

Original dataset with car damage images has 1500 images. This is distributed over 11 classes. To prevent the models from learning to identify the part of the car in the image instead of if damage exists or not, supplementary images are included from the Cars dataset[2]. Figure 1, shows the distribution of the combined training images (9000+) and Figure 2 is sample labelled images. The test set has 9300 images with distribution shown in Figure 1b.

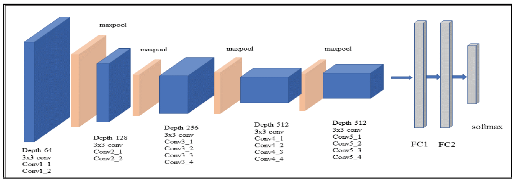
2.2 Models

The project considers seven pre-trained convolutional neural network models for use as base models for transfer learning. The basic structures are explained below.

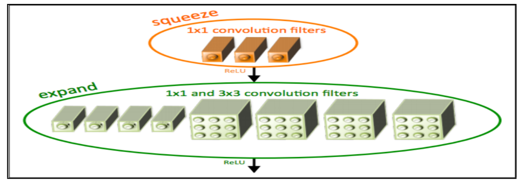
VGG19[3] is a convolutional neural network that is 19 layers deep. The model contains 16 convolutional layers, three fully connected layers and a final layer for softmax function. This was a model that was designed to use smaller 3x3 convolutional structure as compared to previous models like AlexNet.

SqueezeNet[4] stackes a bunch of fire modules and a few pooling layers. The squeeze layer and expand layer as shown in Figure 3b, keep the same feature map size. The squeeze layer reduces the depth to a smaller number and the expand layer increases it.

ResNet(18 and 50)[5] is similar to VGG's full 3×3 convolutional layer design but makes use of residual blocks as shown in Figure 4a (left). The residual block has two 3×3 convolutional layers with

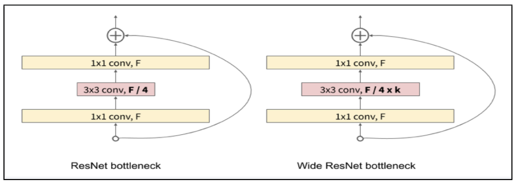


(a) VGG19 model architecture.

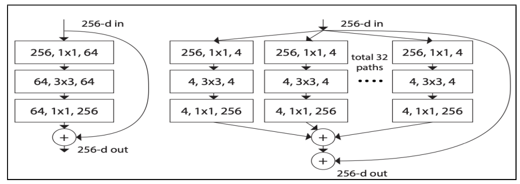


(b) SqueezeNet fire module.

Figure 3: (a) VGG19 model architecture and (b) is the fire module in a SqueezeNet.



(a) Block for ResNet (left) and Wide-ResNet (right).



(b) Block for ResNet (left) and ResNeXt (right).

Figure 4: (a) ResNet vs Wide-ResNet blocks and (b) is ResNet vs ResNeXt blocks.

the same number of output channels. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function.

Wide-ResNet50[6] has increased number of channels compared to ResNet as shown 4a, otherwise the architecture is the same.

ResNeXt101[7] applies a bottleneck design for each block similarly to **ResNet** as shown in Figure 4b. Unlike **ResNet**, within each block there is multiple paths and the number of paths is called the cardinality. In addition, unlike ResNet, in ResNeXt, the neurons at one path will not connected to the neurons at other paths.

DenseNet121 as shown in 5 is a Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with N layers have N connections—one between each layer and its subsequent layer,DenseNet's have $\frac{N(N+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and likewise, that layers feature-maps are used as inputs into all subsequent layers.

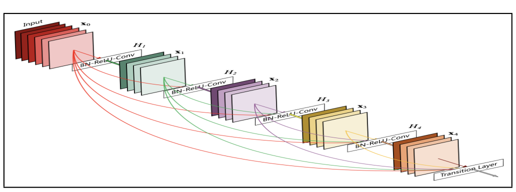
2.3 Training

The models are trained for 25 epochs each using a bath size of 64. The images are transformed as follows:

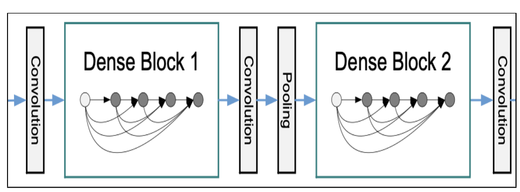
- Random resized crop to 224
- Random horizontal flip
- Normalize with means [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225].

Each of the models above is trained with a configuration taking a combination of:

- Transfer learning type - Fixed features, finetune or hybrid

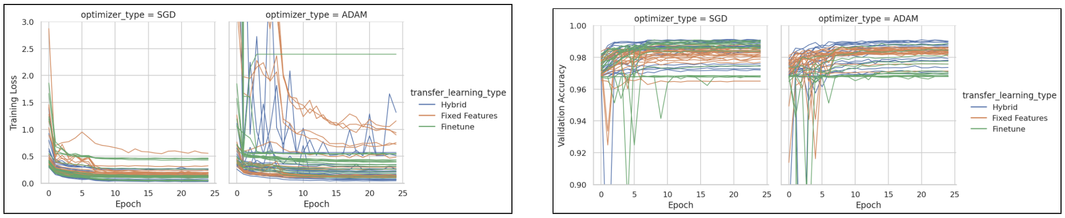


(a) Dense block.



(b) DenseNet structure.

Figure 5: Image distribution by label for training and test set.



(a) Training loss. (b) Validation accuracy.

Figure 6: Training loss (a) and validation accuracy (b) by type of optimizer.

- Optimizer - SGD (with momentum = 0.9) or ADAM
- Initial learning rate - 0.1, 0.01 or 0.001 and scheduled to decay by 0.1 every 7 epochs.

Input: Batched images, pretrained model

Output: New adjusted model, loss, prediction accuracy

Algorithm 1: Pseudo Training Algorithm

for *epoch* $i = 1$ *to 25* **do**
  for *each batch in batched images* **do**
    Forward propagate;
    Make predictions and calculate cross entropy loss;
    Back propagate and update weights which are not deactivated;
  end
end

3 Experiment results

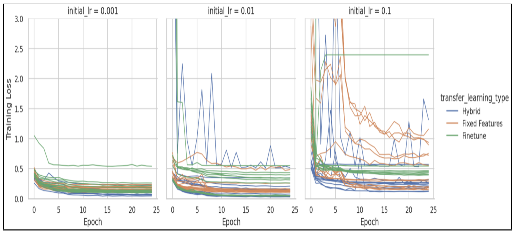
Please provide the results as promised in your proposal to show the "success" of the project. Please provide adequate analysis on how to interpret the result and what conclusion can be inferred. Please also provide implementation details such as hyperparameter choices and datasets etc. (can specify in the appendix if the space is limited). Additional results that are interesting yet not directly relating to your main claim/contribution can be put into the appendix.

3.1 Impact of optimizer on transfer learning performance

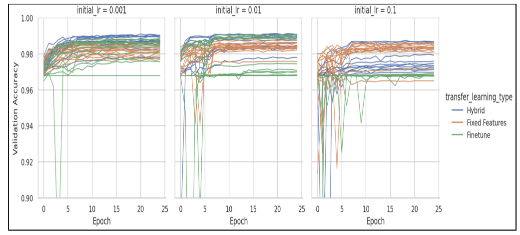
Training loss for SGD in figure 6a converges smoother than when using ADAM. This more noticeable for Hybrid and Fixedfeature models. In terms validation accuracy in figure 6b, the overall top curves are comparable. The distribution of transfer learning types performing well is what seems to be different. For SGD, more Finetuning and Hybrid models are at the top with Fixedfeature models just beneath. On the other hand for ADAM, it seems Fixedfeature models perform comparably to SGD, with Hybrid models clearly outperforming the other transfer learning types. Surprisingly, Finetuned model performance drops significantly, with some models like VGG-19 and SqueezeNet performance dropping as low as 0.003 accuracy.

3.2 Impact of learning rates on transfer learning performance

Training loss in figure 7a decreases in stability for some of the models as learning rate increases. This is more pertinent for Hybrid and Fixedfeature models, but finetuned models seem to still converge smoothly although at a higher loss. In terms validation accuracy in figure 7b, it seems performance declines for Hybrid and Finetuned models as learning increases, whilst Fixedfeature models remain consistent. This is expected as for Fixedfeature models as only the last layer is being retrained.

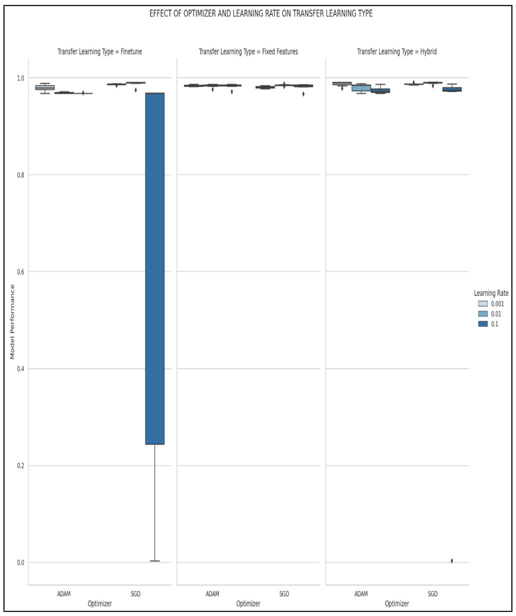


(a) Training loss.

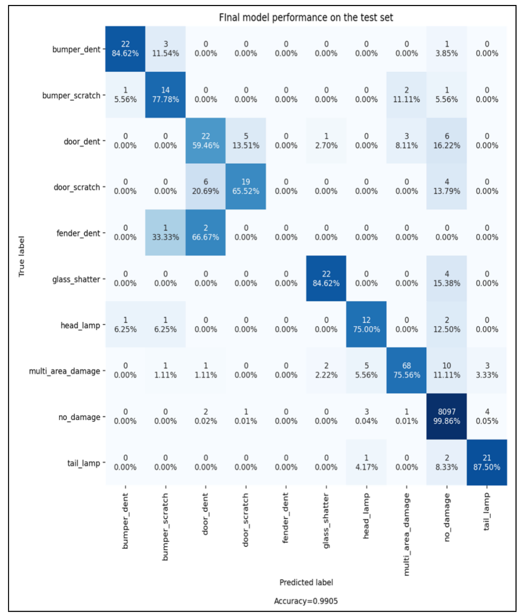


(b) Validation accuracy.

Figure 7: Training loss (a) and validation accuracy (b) by type of initial learning rate (lr).



(a) Boxplot of best validation accuracy for each model.



(b) Final model confusion matrix.

Figure 8: (a) Boxplot of best validation accuracy for each model and (b) is the confusion matrix of the final model performance on test set.

3.3 Overall performance

Figure 8a compares the best test set accuracy achieved by the different transferring learning techniques when using different optimizers and learning rates. This highlights how SGD is sensitive to learning rate when fine tuning of weights is involved, either in **hybrid or finetune** transfer learning. The boxplot shows that the sensitivity increases for SGD as the number of weights being adjusted increases, with the effect more pronounced for the higher initial learning rate 0.1. In addition, the variation in performance is largest for full model fine tuning with learning rate 0.1.

ADAM seems to be robust and less sensitive to learning rate in comparison to SGD. When using pre-trained models as fixed features, models are not as sensitive to type of optimizer and learning rate chosen. Optimal results for finetuning are obtained when training smaller models like ResNet18.

All variations of transfer learning have consistent performance when ADAM is used. SGD with learning rate 0.01 results in best performance in most cases for each technique of transfer learning.

3.4 Final model selection

Table 1 shows the performance on the validation set, of the top 10 models. A Wide ResNet50 based on hybrid transfer learning and trained with SGD and learning rate 0.01, is chosen as the final model based on performance.

Table 1: Top 10 best performing models based on validation accuracy.

Model Architecture	Transfer Learning Type	Optimizer	Learning Rate	Model Performance
Wide-ResNet50	Hybrid	SGD	0.010	0.9913
ResNeXt101-32x8d	Hybrid	SGD	0.010	0.9912
ResNeXt101-32x8d	Hybrid	ADAM	0.001	0.9910
DenseNet121	Finetune	SGD	0.010	0.9909
ResNet50	Hybrid	SGD	0.010	0.9909
ResNet50	Finetune	SGD	0.010	0.9909
DenseNet121	Hybrid	ADAM	0.001	0.9908
ResNeXt101-32x8d	Hybrid	SGD	0.001	0.9903
ResNet50	Hybrid	ADAM	0.001	0.9903
DenseNet121	Hybrid	SGD	0.010	0.9902

3.5 Final model performance

Figure(8b) is the confusion matrix for the final model, with the accuracy normalized by row. This highlights how the model performs for each class label of damage as overall accuracy will not capture all facets of performance by the models. The model performs satisfactorily in identifying most classes except fender dents, which have insufficient samples in the training set and only 3 images for testing. For the damaged cars, the model performs best in identying damage on tail lamps, glass shatter and bumper dents, as this all have class accuracy's above 0.8. More data is required to first test fender damage identification and if that maintains the poor performance, then retraining might be required.

4 Conclusions

In conclusion, the project managed to produce a reasonable model which has the ability to determine existence of damage as well as location and to an extent severity. In addition, key factors influencing performance of different types transfer learning techniques for this kind of task are identified. From the results, it is observed that if SGD is chosen as an optimizer for a transfer learning task, then care should be taken, as the potential results could have high variance. A good choice for an initial learning rate for SGD for this task is 0.01, as 0.1 is deemed to be too high.

In terms of model choice, a few of the models could have been suitable candidates, but a wide resnet model trained with SGD and learning rate 0.01 is chosen.

Including the supplementary images allowed the model to differentiate damaged from non-damaged cars, but has also introduced significant class imbalances. Further work, will be to consider sample balancing techniques like making use of transformations to generate more data. In addition, other feature descriptor techniques like histogram of oriented gradients (HOG) will be explored. Post that, work on object detection will be done to enhance project applicability to the insurance case.

References

[1] Srimal Jayawardena. *Image Based Automatic Vehicle Damage Detection*. PhD thesis, 11 2013.

[2] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[4] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.

[7] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.