VEHICLE DAMAGE ANALYSER

DECLARATION BY THE CANDIDATES

We the undersigned solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of _____. We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that :

I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/ diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

PREPARED FOR

IBM Internship Project

PREPARED BY

Anirudh Muhnot
Ananya Bhatnagar
Gunjan Agrawal
Kajal Gupta
Rohit Patidar
Vedant Sharma

Project Overview

1. Project Abstract

Vehicle damage identification of vehicles is a challenging problem primarily handled with extensive training data and expertise. We've trained a vision classifier with the goal of identifying the damage in vehicle as accurately and precisely as possible. Our dataset consists of images for 5 different vehicle damage categories. Additionally, we perform multiple experiments to explore and understand what are the backend processes that might be running in our API calls. Our results demonstrate that transfer learning with Resnet-34 architecture are a powerful tool to classify images for our problem.

2. Project Plan

Being unaware of any past implementations of Resnets in the context of vehicle damage identification, it was difficult to predict a prior how far we could get within this research. We therefore decided to divide the damage classification process into multiple steps, so that we can start with a relatively easy task and increase complexity when we progress. That is, we will first develop a method to classify whether the vehicle is damaged or not. Since the dataset also includes images of many different types of vehicles and transfer learning performs well here, we expect this first task to be a good first approach examine our methods. After that, we proceed to our main task, which is to assess our model's performance.. Since damages may look very different depending on the type, location and severity of the damage, we expect this task to be much harder than the first one.

3. Contents

3.1) **Tools Used**

- Python
- Frameworks Used
    - Dash by Plotly
    - MaterializeCSS

- FAST.AI

3.2) **Project Implementation**

3.2.1. Introduction

Today, in the car insurance industry, a lot of money is wasted due to claims leakage. Claims leakage / Underwriting leakage is defined as the difference between the actual claim payment made and the amount that should have been paid if all industry leading practices were applied. Visual inspection and validation have been used to reduce such effects. However, they introduce delays in the claim processing. There have been efforts by a few start-ups to mitigate claim processing time. An automated system for the car insurance claim processing is a need of the hour.

Here, we employ **transfer learning from IMAGENET trained RESNET-34 model** for classification of car damage types. Specifically, we consider common damage types such as broken windshield, bumper damage, broken lights and flat tyre. To the best of our knowledge, there is no publicly available dataset for car damage classification. Therefore, we created our own dataset by collecting images from web and manually annotating them. ***The classification task is challenging due to factors such as large inter-class similarity and visible damages.*** We experimented with many techniques such as directly training a CNN, using from large CNNs trained on **Xception, then eventual moving to Resnet34**. Experimental results validate the effectiveness of our proposed solution.

3.2.2. Dataset Description

Since there is no publicly available dataset for car damage classification, we created our own dataset consisting of images belonging to different types of car damage. We consider seven commonly damage types such as broken windshield, bumper damage, broken lights and flat tyre. In addition, we also collected images which belong to a no damage class(positives). The images were collected from web and were manually annotated.

Regarding our stepwise classification procedure, we need to obtain two datasets, containing respectively with undamaged vehicles and with damaged vehicles. In the following paragraphs, we describe the **collected datasets.**

Following the method of Griffin et al. (2007) for creating Caltech-256, we ‘scraped’ Google Images (using Python) and downloaded all images yielding from different queries.

Images with undamaged cars. This category contains images of cars with no damage. These images are biased towards cars which are not damaged as a whole with images from 4 other categories are less in number. This is due to the fact that our classifier was able to correctly classify features in damaged vehicles but was not able to classify cars with no damage when the whole vehicle was shown.

Images with damaged cars. No dataset of images with damaged cars has been found, so we needed to create our own dataset here. For the diversity of our dataset, we ensured that the obtained car damages are of different types and severities. We manually checked all collected images and deleted the inapplicable or duplicate ones.

3.3.3 Related Works

1. Resnet34

The problem of training very deep networks has been alleviated with the introduction of a new neural network layer — **The Residual Block. (Fig 2)** The core idea of ResNet is

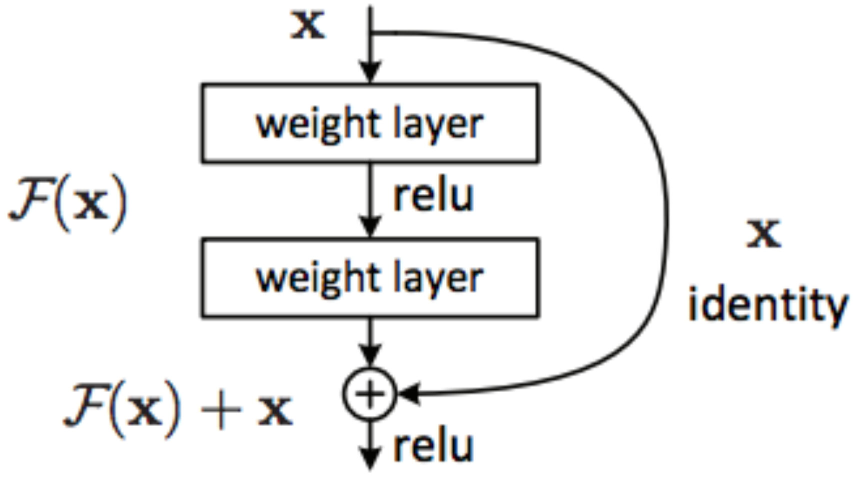
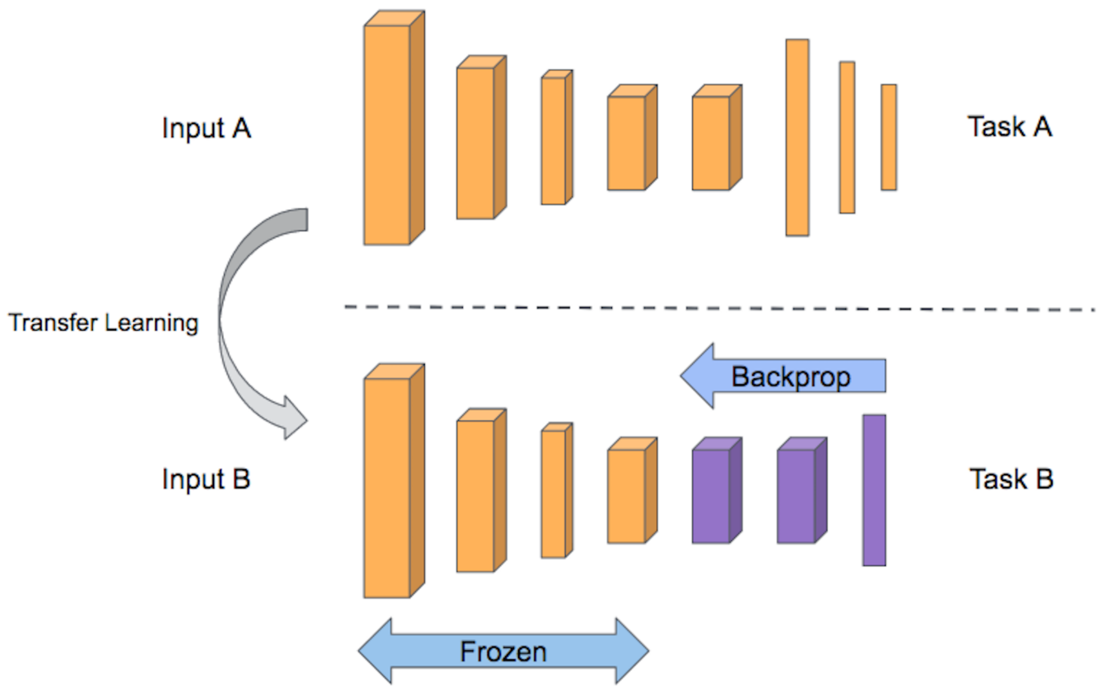

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

introducing a so-called “identity shortcut connection” that skips one or more layers. Sacking layers shouldn’t degrade the network performance, because we could simply stack identity mappings (layer that doesn’t do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. Letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlaying mapping. And the residual block above explicitly allows it to do precisely that.

2. Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

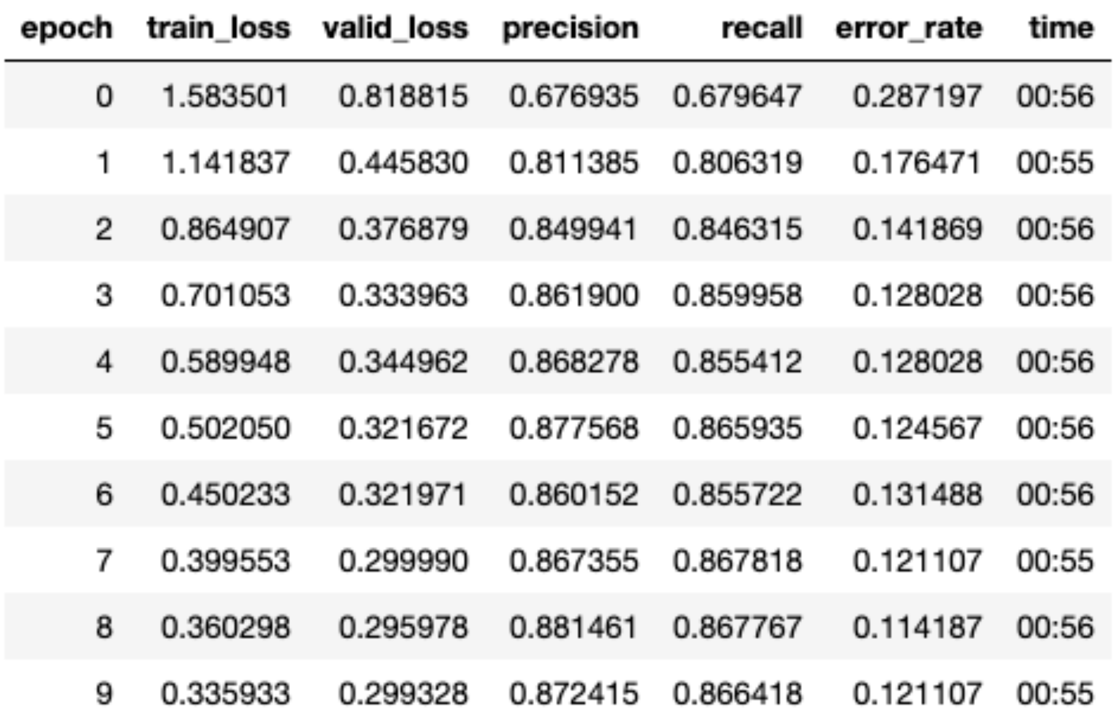


Pre-trained Model Approach:

1. Select Source Model. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
2. Reuse Model. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
3. Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

3.3. Training in FAST AI

Fast.ai is one organization that tries to address this inequity. It was founded in 2016 by Jeremy Howard and Rachel Thomas, with the goal of making deep learning more accessible. It’s primarily known for its free courses and open source library (Named fastai and built on top of Facebook’s PyTorchlibrary). Fast AI makes deep learning simple and with high accuracy. It has a vision module that contains a CNN_learner function which is, in turn, used to create a transfer learning Resnet-34 model. Here is how the model performed on the first 10 epochs:

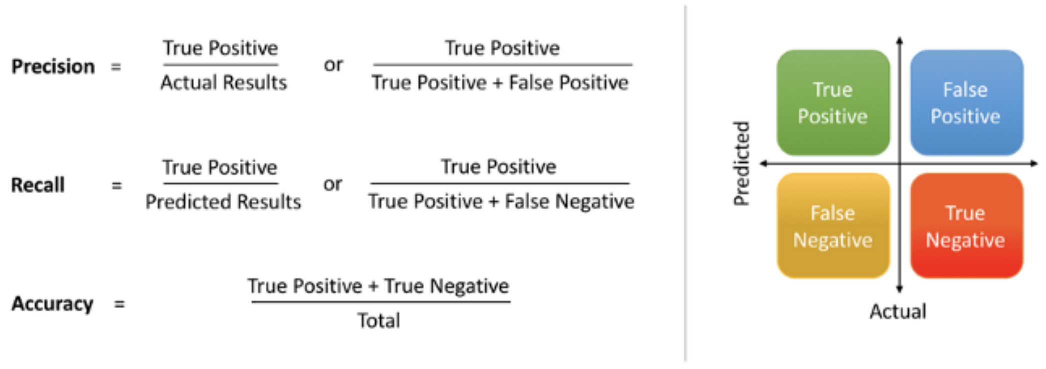


epoch	train_loss	valid_loss	precision	recall	error_rate	time
0	1.583501	0.818815	0.676935	0.679647	0.287197	00:56
1	1.141837	0.445830	0.811385	0.806319	0.176471	00:55
2	0.864907	0.376879	0.849941	0.846315	0.141869	00:56
3	0.701053	0.333963	0.861900	0.859958	0.128028	00:56
4	0.589948	0.344962	0.868278	0.855412	0.128028	00:56
5	0.502050	0.321672	0.877568	0.865935	0.124567	00:56
6	0.450233	0.321971	0.860152	0.855722	0.131488	00:56
7	0.399553	0.299990	0.867355	0.867818	0.121107	00:55
8	0.360298	0.295978	0.881461	0.867767	0.114187	00:56
9	0.335933	0.299328	0.872415	0.866418	0.121107	00:55

3.4. Error Metrics:

Precision and Recall:

Precision means the percentage of your results which are relevant. On the other hand, recall refers to the percentage of total relevant results correctly classified by your algorithm.



$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

3.5. Results

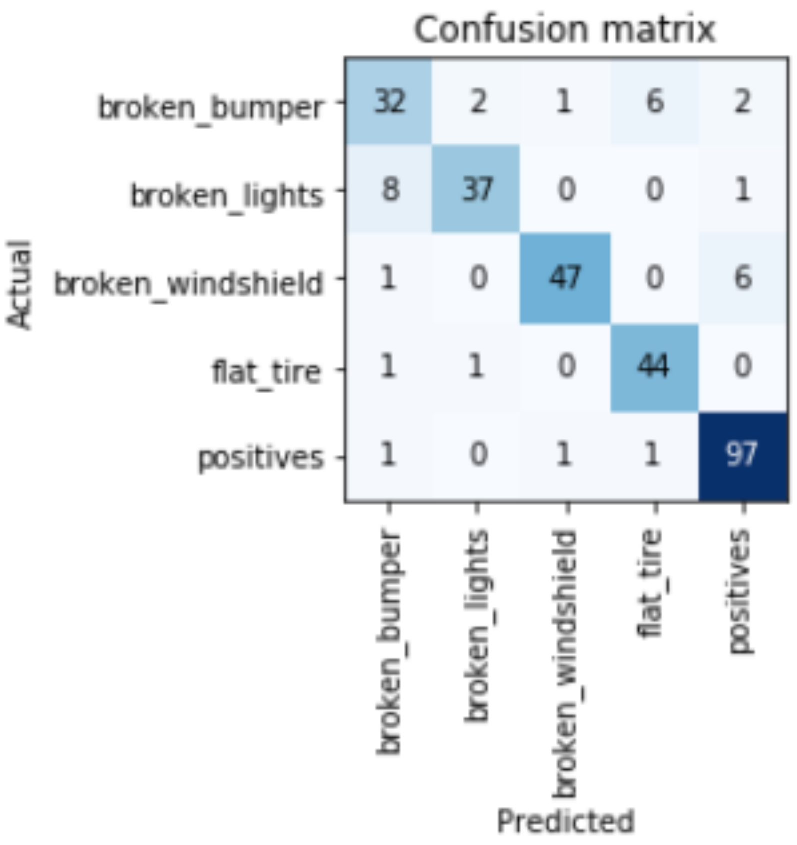
We were able to achieve 86% validation recall and validation on a training dataset of 1156 images with an error rate of 12% overall.

Finally, when unfreezing the starting layers we were able to achieve the accuracy of 89% with increased accuracy and recall.

epoch	train_loss	valid_loss	precision	recall	error_rate	time
0	0.270039	0.287886	0.866977	0.858427	0.121107	00:56
1	0.232671	0.295970	0.881242	0.869085	0.110727	00:57

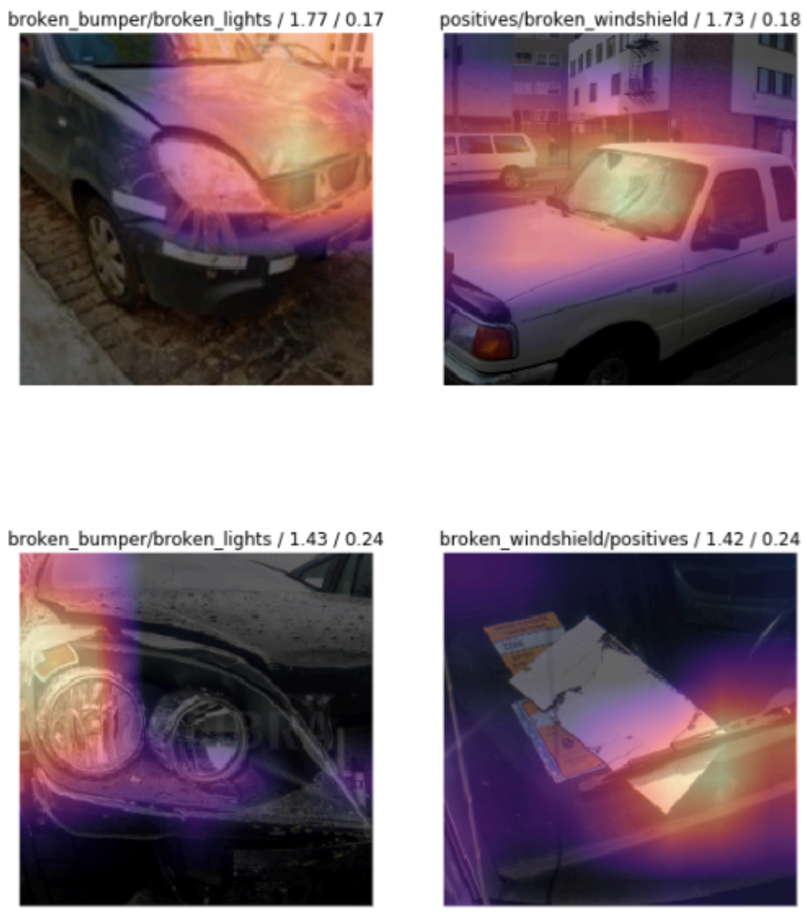
3.6 Errors

Overall breakdown of errors on the validation set is as follows:



Actual \ Predicted	broken_bumper	broken_lights	broken_windshield	flat_tire	positives
broken_bumper	32	2	1	6	2
broken_lights	8	37	0	0	1
broken_windshield	1	0	47	0	6
flat_tire	1	1	0	44	0
positives	1	0	1	1	97

The classes broken_lights and broken_bumpers have interclass-similarities so they have the most miss-classified examples. Also, some errors occur due to arbitrary watermarks in the dataset. Top losses in the classification are as follows:

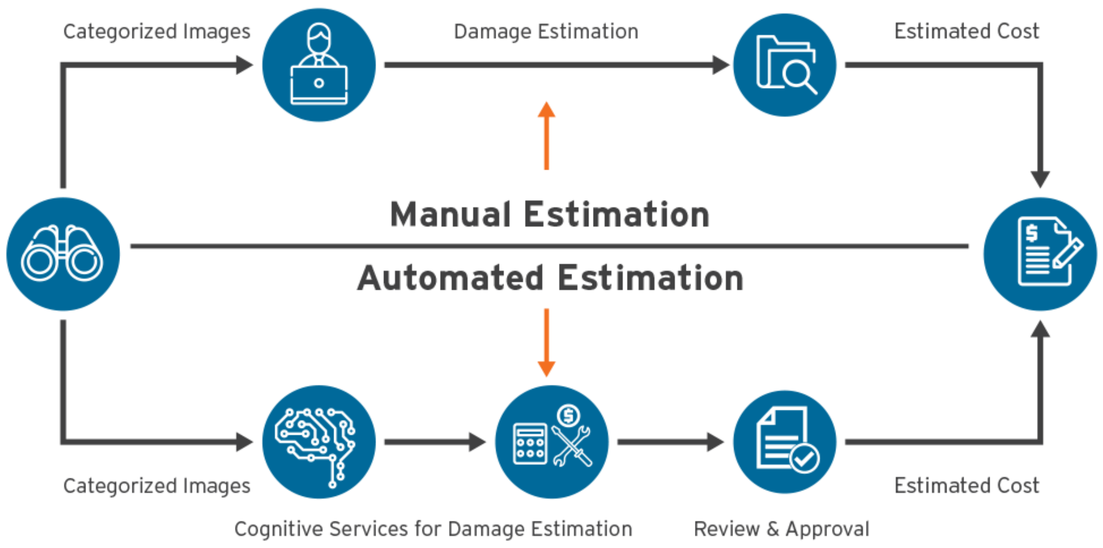

broken_bumper/broken_lights / 1.77 / 0.17
positives/broken_windshield / 1.73 / 0.18
broken_bumper/broken_lights / 1.43 / 0.24
broken_windshield/positives / 1.42 / 0.24

3.6) **Use Cases**

Vehicle insurance claims involve a manual process with expertise needed from domain specialists to evaluate the validity of the claims and their adjudication. Further, there are fraudulent claims that need to be identified, which results in inefficient usage of time for adjudication, and in some cases in fraudulent claims being approved. This manual process is not cost-effective, and fraudulent claims adds to the expense. It is also time consuming, hence there is a need to expedite the claims settlement process for such claims.

An image processing system and method obtains source images in which a damaged vehicle is represented, and performs image processing techniques to determine, predict, estimate, and detect damage that has occurred. The image processing techniques may include generating a composite image of the damaged vehicle, aligning and isolating the image, applying convolutional neural network techniques to the image to generate damage parameter values, where each value corresponds to damage in a particular location of vehicle, and other techniques.

Our solution uses a blend of machine learning expertise and database solutions to create an application which involves a site visit where the agent can upload images of the damaged vehicles, with the claims details. These images are then fed through a machine learning pipeline which identifies the damaged parts of the vehicle, and estimates the cost of repair or replacement for the damaged parts.



4. Conclusion

In this project, we proposed a deep learning based solution for car damage classification. Since there was no publicly available dataset, we created a new dataset by collecting images from web and manually classifying them. We experimented with multiple deep learning based techniques such as training CNNs from random initialisation and testing out various pre-trained models such as ResNet34, Xception, InceptionV3 etc. We observed that the transfer learning performed the best. We also note that car specific features are effective for damage classification. It thus underlines the superiority of feature representation learned from the large training set. Further steps to increase the Accuracy can be done by increased by using a memory intensive architecture such as ResNet-50 and training the model for more time with a larger dataset.

5. References

- https://keras.io/applications/
- https://www.analyticsvidhya.com/blog/2018/07/building-mask-r-cnn-model-detecting-damage-cars-python/
- http://www.freepatentsonline.com/9886771.html
- http://cs231n.stanford.edu/reports/2017/pdfs/406.pdf
- https://www.fast.ai/