Premise - To propose the build of an algorithm to identify damaged areas of the vehicle. Images can contain:

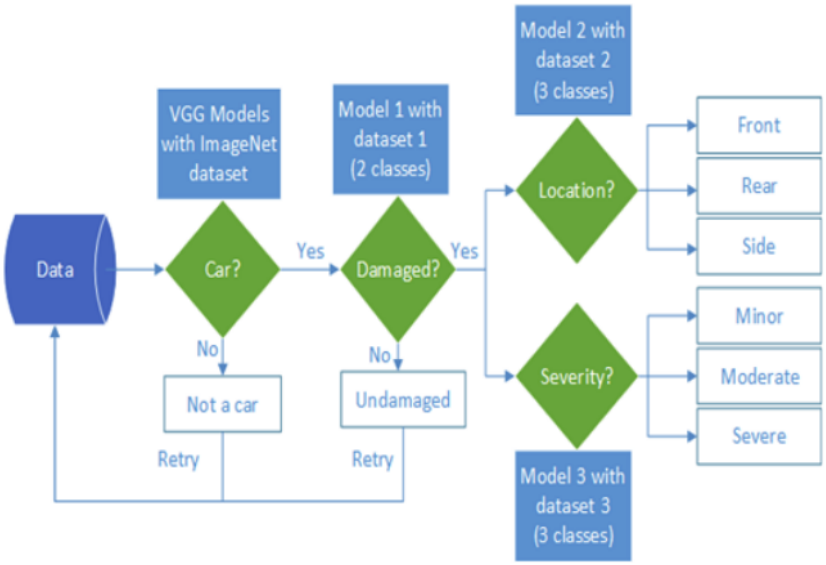
- No vehicles
- One Vehicle
- Multiple Vehicles

What are the tradeoffs between transfer learning vs deep learning for cases like these?

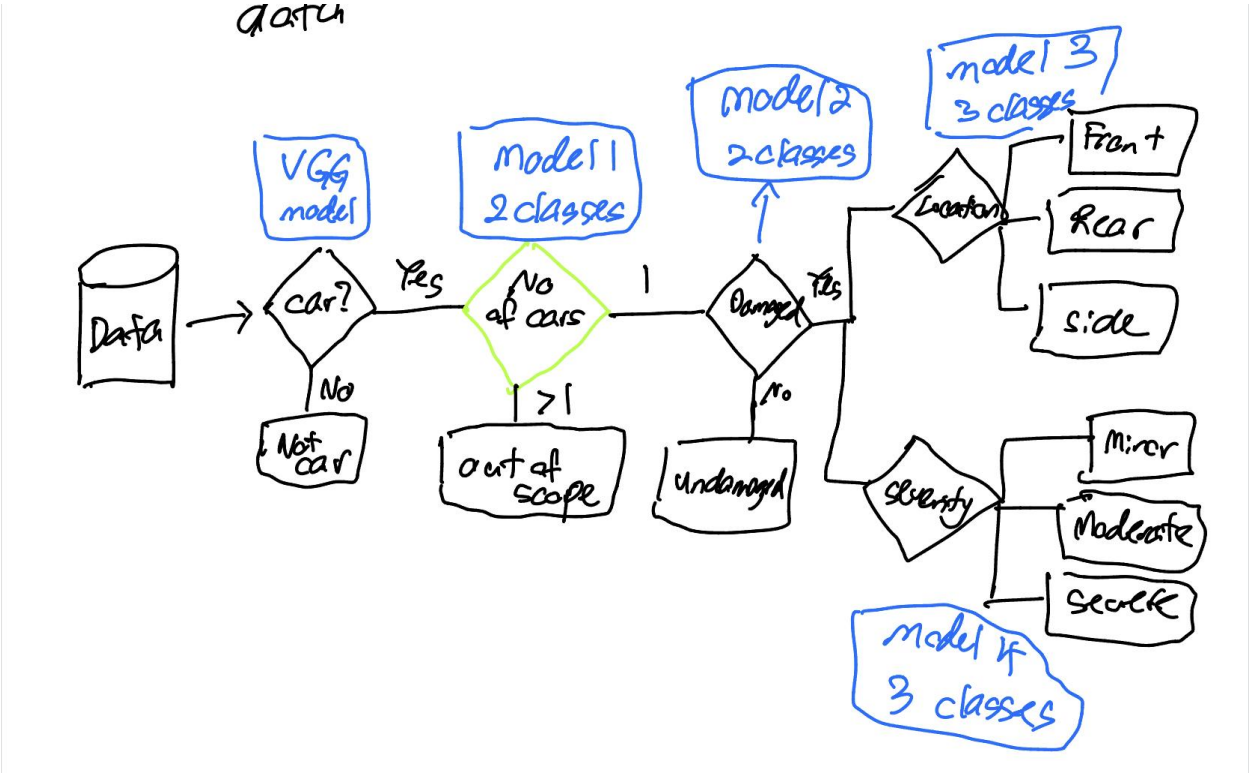
- Assuming that “deep learning” in this case refers to training a completely new neural network just to identify cars/identify damage
- Transfer learning in this case would require less data, leveraging on deep learning models with pre-trained weights instead
- Unless we have a lot of data to train on, transfer learning should be the case for computer vision tasks (https://ruder.io/transfer-learning/index.html#usingpretrainedcnnfeatures)
- However, transfer learning requires that a strong correlation between the knowledge of the pre-trained source model and the target task domain for them to be compatible (https://www.v7labs.com/blog/transfer-learning-guide)
- Hence, good options are to choose the pre-trained VGG models of VGG16 and VGG19 trained on the ImageNet dataset (Car Damage Assessment Based on VGG Models https://site.ieee.org/thailand-cis/files/2019/12/JSCI8-Paper-5.pdf)

<u>Proposed number of models</u>

Referencing the the report made by Phyu and Kuntpong, 2019 (https://site.ieee.org/thailand-cis/files/2019/12/JSCI8-Paper-5.pdf), a proposed architecture would be made up of one that is similar to the below, with the exception of adding another model for the purpose of detecting the ***number of cars*** involved in the image.



A slightly modified version of the flowchart will be as below:



With the inclusion of the new model to detect the number of cars in the image, a scope was made on the number of cars that the pipeline will accept. For now, the assumption is made that damage will be detected with 1 car.

A short description on how Model 1 is to be trained, with goal of detecting a binary classification of 1 car or > 1 car.

- Obtain images of 1 car , and more than 1 car
- Using keras, create a numpy iterator, specifying the batch size, resizing images to fit a certain size
- Preprocessing - Turning the images from 0, 255 colours pixel values to 0, 1, decreasing burden on neural network processing
- Data splitting for train, validation, and test
- Instantiate a Keras sequential model
    - First convolution layer - 16 filters, 3*3 pixels, and stride 1, relu activation function
    - Max pooling layer - 2*2 square from output of first convolution layer
    - Convolution layer with 32 filters
    - Convolution layer with 16 layers
    - Flatten layer
    - Final layer with 1 output