

# Capstone Project

## Machine Learning Engineer Nanodegree

Rafael Castillo Alcibar

September 15th, 2016

### I. Definition

#### Project Overview

This project is a proof of concept (POC) solution where deep learning techniques are applied to vehicle recognition tasks, this is particularly important task in the area of traffic control and management, for example, companies operating road tolls to detect fraud actions since different fees are applied with regards to vehicle types. Images used to train neural nets are obtained from the [Imagenet](#) dataset, which publicly distributes images URLs for hundreds of categories. Since the whole experiment is performed on a personal computer with limited computational resources, POC scope is also limited to the simple classification of two different kinds of vehicles: Trailer Trucks versus Sports Cars. Main POC's goal is to determine the maximum accuracy (percent of times model was correct on its predictions) different neural nets with basic architectures can reach using a limited set of images (less than 700) for training.



*Illustration 1: Examples of input images for each class.*

The whole project is constructed using [Keras](#), which is a highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano and it was developed with a focus on enabling fast experimentation. In this case, Theano is the backend selected.

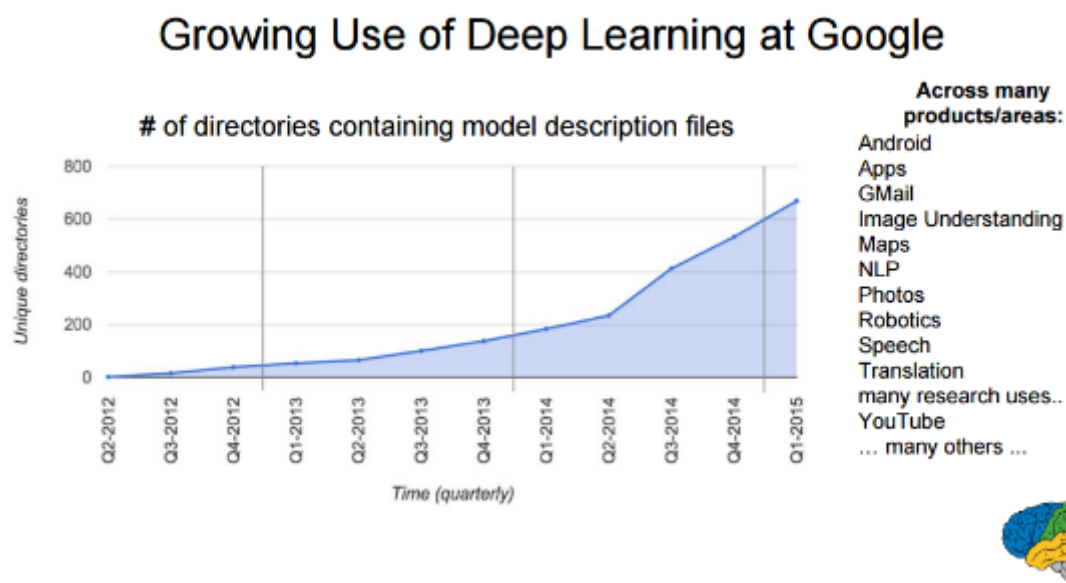
#### Some Deep Learning Background

The first general, working learning algorithm for supervised deep feedforward multilayer perceptrons was published by Ivakhnenko and Lapa in 1965. In 1989, Yann LeCun et al. were able to apply the standard backpropagation algorithm, which had been around as the reverse mode of automatic differentiation since 1970 to a deep neural network with the purpose of recognizing handwritten ZIP codes on mail. Despite the success of applying the algorithm, the time to train the network on this dataset was approximately 3 days, making it impractical for general use. According to LeCun, in the early 2000s, in an industrial application CNNs already processed an estimated 10% to 20% of all the checks written in the US in the early 2000s. The significant additional impact of deep learning in image or object recognition was felt in the years 2011-2012. Although CNNs trained by backpropagation had been around for decades, fast implementations of

CNNs with max-pooling on GPUs were needed to make a dent in computer vision. In 2011, this approach achieved for the first time superhuman performance in a visual pattern recognition contest.

Deep learning is often presented as a step towards realizing strong AI and thus many organizations have become interested in its use for particular applications. In December 2013, Facebook hired Yann LeCun to head its new artificial intelligence (AI) lab. The AI lab will develop deep learning techniques to help Facebook do tasks such as automatically tagging uploaded pictures with the names of the people in them.

In 2014, Google also bought DeepMind Technologies, a British start-up that developed a system capable of learning how to play Atari video games using only raw pixels as data input. In 2015 they demonstrated AlphaGo system which achieved one of the long-standing "grand challenges" of AI by learning the game of Go well enough to beat a human professional Go player. [ref 04]



*Illustration 2: Growth of the Deep Learning techniques in Google in the past 3 years*

## Problem Statement

For this project, it is used a personal computer (Intel® Core™ i5-4310M CPU @ 2.70GHz × 4, 8 Gb RAM and 64-bit). Different deep learning models are trained and validated and their results compared in order to determine which architecture maximizes prediction scores in the vehicle classification recognition while minimizing computational costs. Vehicles images will be used to train and test models while a subset of images are used for validation (these are unseen images for models not previously used during the train/test phase).

## Metrics

Since this is a binary classification problem, basically model will try to respond to the question: "Is the vehicle of this picture a trailer truck or a sports car?", scores like Precision, Recall, F-Scores or Accuracy are suitable. For simplicity, and since the dataset is balanced (there are a similar number of images for each class), accuracy is the score used to evaluate models performances. Accuracy gives an estimate of how an often model is correct on its predictions, that is, how often model correctly flags a truck like a truck and a sports car as a sports car.

In the other hand, since computational resources is also a critical point to consider, minutes required to train the model is the second metric used. Combination of both scores allows the identification of the model that maximizes precision while minimizing computational resources.

## II. Analysis

### Data Exploration

Images are collected from Imagenet dataset which contains hundreds of different categories, [synsets](#) is used to identify any particular category, for example, in this particular case the following synsets are used:

- 1.n04467665: Trailer Trucks
- 2.n04285008: Sport-Cars

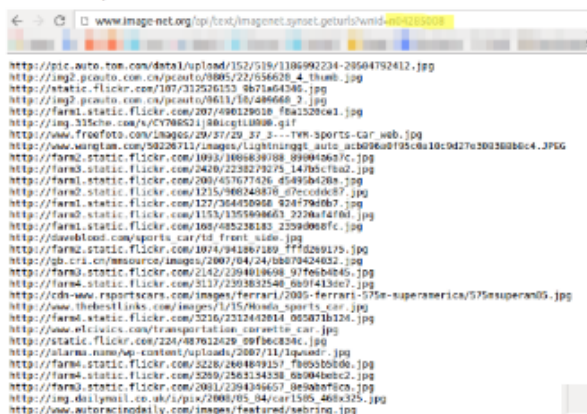
To retrieve images, since image URLs are freely available, the process to download by HTTP protocol requires a little python script to download pictures from urls listed in the following link:













[http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=\[wnid\]](http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=[wnid])

where `[wnid]` is one of the synsets selected previously. For further reference, please check [Imagenet documentation](#).

Synsets:

1. n04467665: Trailer Trucks
2. n04285008: Sport-Cars

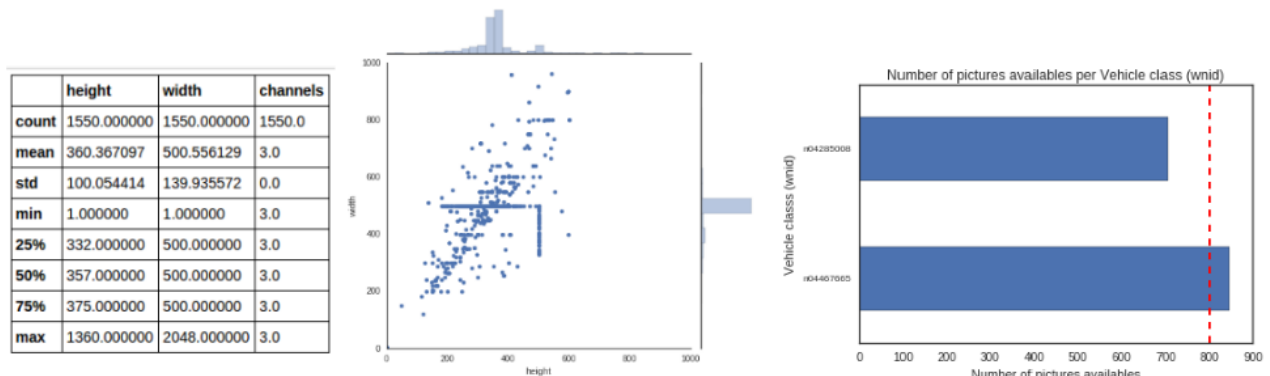



n04285008		DeepLearning			
Name	Size	Type	Modified		
 n04285008_2.jpg	143,5 kB	Image	sep 6		
 n04285008_9.jpg	151,4 kB	Image	sep 6		
 n04285008_13.jpg	148,5 kB	Image	sep 6		
 n04285008_15.jpg	59,3 kB	Image	sep 6		
 n04285008_18.jpg	107,6 kB	Image	sep 6		
 n04285008_23.jpg	34,9 kB	Image	sep 6		
 n04285008_27.jpg	168,8 kB	Image	sep 6		
 n04285008_33.jpg	142,0 kB	Image	sep 6		
 n04285008_35.jpg	49,2 kB	Image	sep 6		
 n04285008_40.jpg	109,1 kB	Image	sep 6		
 n04285008_50.jpg	62,5 kB	Image	sep 6		
 n04285008_51.jpg	109,2 kB	Image	sep 6		

*Illustration 3: Summary of the download process*

Image below summarizes main dataset characteristics, a total of 1550 pictures are available with a mean height and width of 352 and 483 pixels respectively with 3 channels (RGB) for colors.

Since there are pictures with 1px height and width, pictures with less than 150px for each dimension are removed since are considered to not be valid for the project due to their poor resolution, this process eliminated just 32 images.



*Illustration 4: Distribution and characteristic of the downloaded data*

With regards of the different classes, just one of them include more than 800 pictures, since the dataset is at this stage unbalanced and small, [Keras Data Generator](#) utility is employed to generate fake images from the pictures already available, for example, the picture below. This utility is crucial to balance classes and generate new images to train models [ref 05].



*Illustration 5: Data augmentation example*

## Exploratory Visualization

In order to use images as input for the deep learning models, images need to be converted into multidimensional arrays of number where each pixel represents a cell in the multidimensional array. For this process it is used the Numpy library `ndimageas` described in this [tutorial](#). In this project images are resized to 150px height and width respectively in gray scale of colors (since color is not a determinant characteristic to differentiate between a truck and a sports car) which reduces dimensions in two dimensions (RGB = 3 channels, grayscale = 1 channel)

```

208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241
242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277
278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357
358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397
398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437
438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597
598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637
638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677
678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717
718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757
758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797
798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837
838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877
878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920
921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961
962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

```



Illustration 6: Images are represented as matrix where a pixel is a number in the range 0-255

# Algorithms and Techniques

Different deep convolutional neural nets architectures are used to perform this task, which nowadays seems to be the best known approach in the image recognition field. Images categorization is a complex task, for example, a grayscale image of size 150x150 would be transformed to a vector of size  $150 \cdot 150 = 22500$  for a fully connected neural network. Such huge dimensionality with no predefined features makes this problem unapproachable for standard supervised learning approaches, even combining them with dimensional reduction techniques like PCA.

Convolutional nets are elected to be the most efficient technique to extract relevant information from, in this case, images to be used in classification tasks. When used for image recognition, convolutional neural networks (CNNs) consist of multiple layers of small kernels which process portions of the input image, called receptive fields. [This blog by Victor Powell](#) is an excellent resource to understand how kernels works. Kernels are small matrix (normally 3x3 or 5x5) applied over the input image to extract features from data, this technique has been used in image processing for decades, from Photoshop filters to medical imaging.

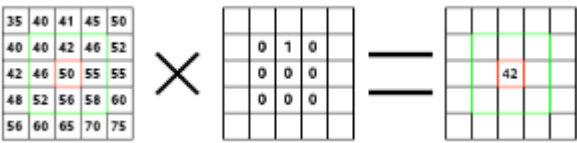
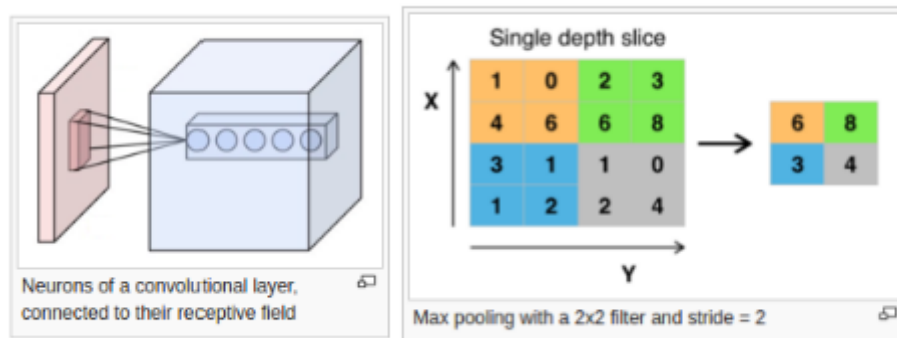


Illustration 7: How kernel works. A kernel matrix is applied to the input image to generate an output. This matrix is applied to every pixel in the input.

The outputs of these kernels are then tiled so that their input regions overlap, to obtain a better representation of the original image; this is repeated for every such layer. Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters. Compared to other image classification algorithms, convolutional neural networks use relatively little pre-processing. This means that the network is responsible for learning the filters that in traditional algorithms were hand-engineered. The lack of dependence on prior knowledge and human effort in designing features is a major advantage for CNNs.

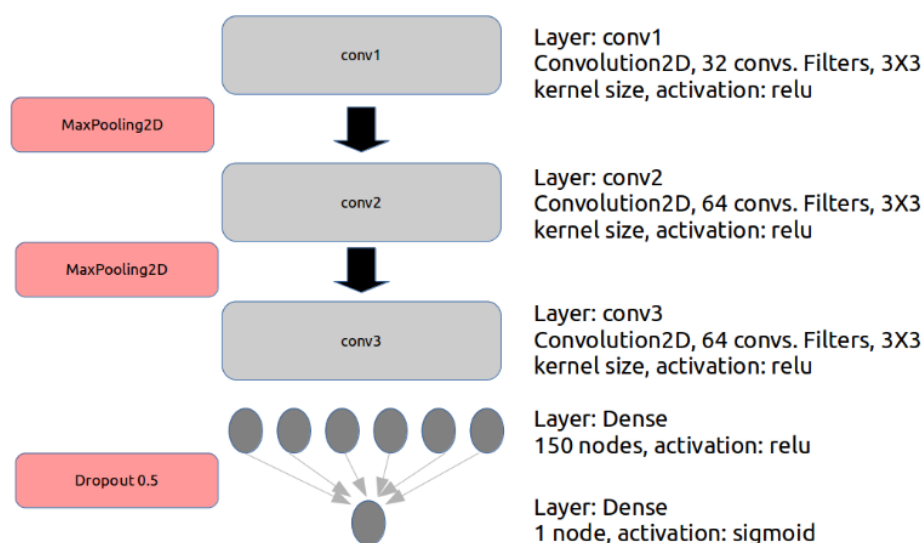


Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer in-between successive conv layers in a CNN architecture. The pooling operation provides a form of translation invariance.[ref 06].



*Illustration 8: CNNs representation and MaxPooling*

The proposed net architecture for this particular problem is a neural net with 1 to 4 layers where each layer includes a CNN + Max Pooling layer. On top of that it is included a fully connected net with 150 nodes in the input side and 1 node to output results and dropout implemented. Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data, it basically consist in dropping out nodes randomly in a neural network to gain robustness in model predictions. Below is included an example of a proposed architecture:



*Illustration 9: Neural network architecture*

## Benchmark

In the study: [Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning](#) several machine learning approaches are used for car detection and identification. A fine-grained dataset containing 196 different classes of cars is employed. This dataset is particularly challenging due to the freeform nature of the images, which contained cars in many different sizes, shapes, and poses, similar scenario applies to the current dataset, but in this particular case there are just two different classes. Study results are presented in terms of accuracy for the top1 and top5 classes for the different approaches used. For the Deep Learning approaches, accuracy values are around 0.8, so this will be the value used to benchmark current results.

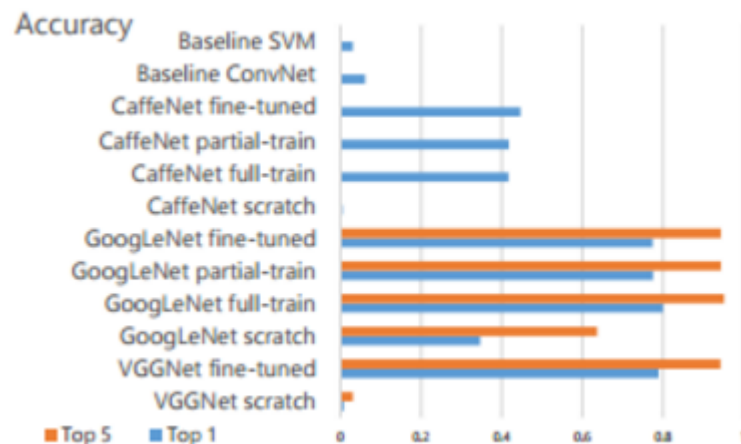


Illustration 10: Summary results for the different models attempted.

## III. Methodology

### Data Preprocessing

Vehicles images are downloaded from the Imagenet dataset, in the notebook included is described the process step-by-step, basically it is required to include the configuration parameters in `configuration.py` and executed `download_imagenet_images.py`.

Once all files are downloaded, different picture classes needs to be organized following the structure:

```
dataset\
  train\
    n04467665\
      n04467665_01.png
      n04467665_04.png
      ...
    n04285008\
```

```
n04285008_01.png
n04285008_04.png
...
test\
n04467665\
n04467665_02.png
n04467665_03.png
...
n04285008\
n04285008_02.png
n04285008_03.png
...
validation\
n04467665\
n04467665_07.png
n04467665_09.png
...
n04285008\
n04285008_07.png
n04285008_09.png
...
```

For this purpose, `Data_Wrangling.py` is employed. Next step is to eliminate those pictures with height and width lower than 150px. A threshold of 150px is employed since this is the images dimensions used in common Deep Learning models nowadays and the size of the input images for the models. To end, `ImageProcessor.py` is employed to perform several tasks:

1. Generate augmented images from current images using the Keras utility `ImageDataGenerator` to be used to train models.
  - `rescale=1./255`: as the images taken by Raspberry Pi's camera come with RGB coefficients in the range of 0-255 I had to normalize the values to span from 0 to 1., which was achieved by this scaling
  - `rotation_range=40`: images were rotated randomly by 0-40 degrees
  - `width_shift_range=0.01`: range in which image was randomly translated vertically



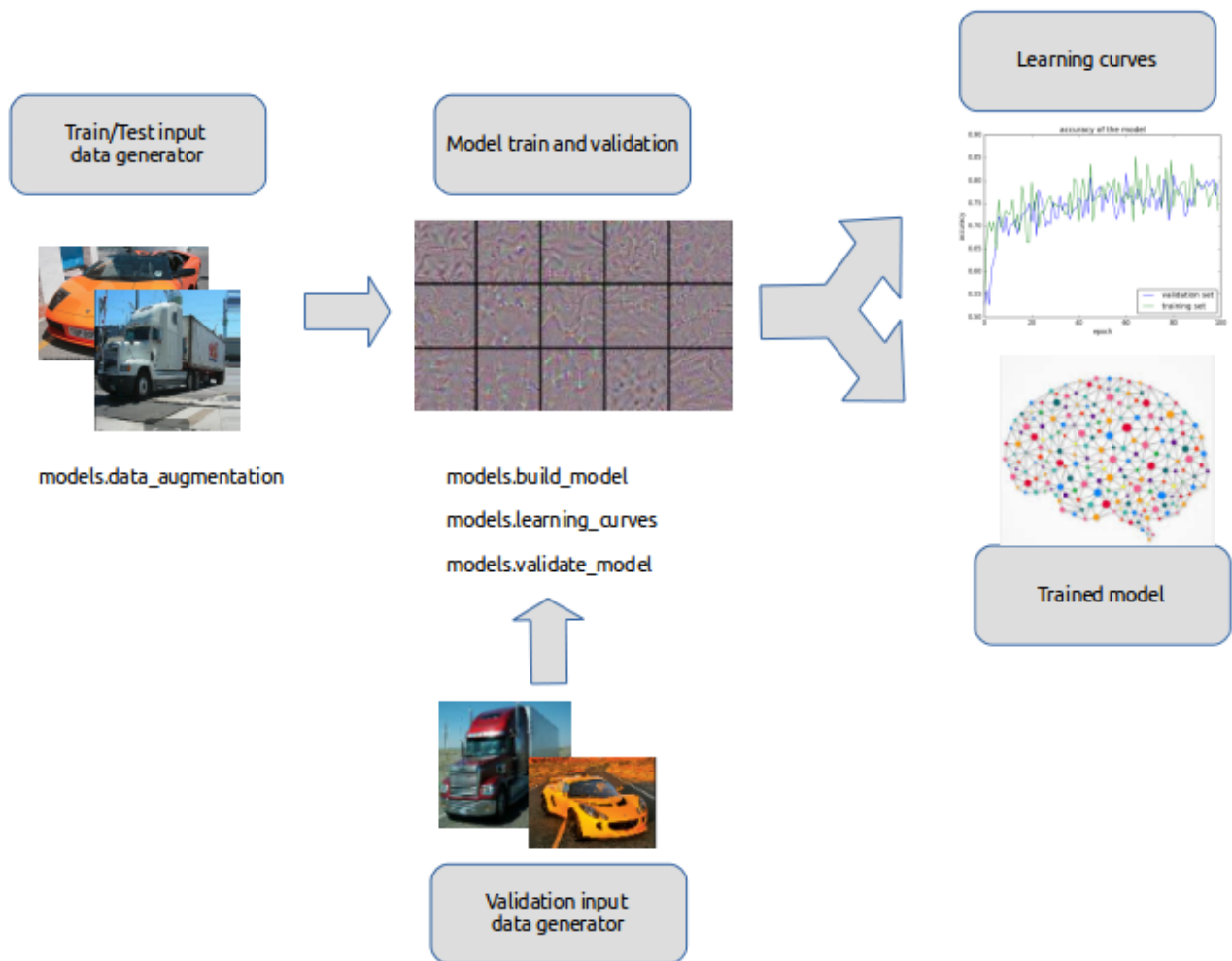
- height\_shift\_range=0.1: range in which image was randomly translated horizontally
  - shear\_range=0.05: range in which shearing transformations were applied randomly
  - zoom\_range=0.1: range in which image was zoomed at randomly
  - fill\_mode='nearest': this was the method with which newly introduced pixels were filled out
- 2.Resize pictures to height and width of 150px.
  - 3.Use a gray scale for the picture colors (since color is not a important feature to distinguish a truck from a car).

## Implementation

For the implementation I have chosen Keras. Keras is a neural network library for Theano and TensorFlow written in Python. Different convolutional neural net architectures were applied for the task with the intention of identifying the architecture that reached a reasonable accuracy with the minimum computational resources. Networks consisted of an input layer, 1 to 4 convolutional layers, a fully connected layer, and an output layer. The convolutional layers used 3x3 convolutions and 32-64 output filters followed by max pooling layers of 2x2. For the activation functions rectified linear units are used, except for the final output neuron which was sigmoid. After the fully connected layer a dropout of 0.5 was applied (this helps to prevent overfitting). For the loss function I have used logloss (binary crossentropy). Two different optimizers are used and compared: `adam` and `Adagrad`.

With regards to the difficulties encountered in the process, the first difficulty was to understand how data augmentation should be carried out. It needed a bit of trial and error to figure out how much I can distort the images, such that the car/truck remains on the images all the time. A further difficult was to figure out how to visualize the filters. Although I had a [solution to start from](#), the code needed some workaround, such as referencing my convolutional layers, writing a function to draw the images for all filters.

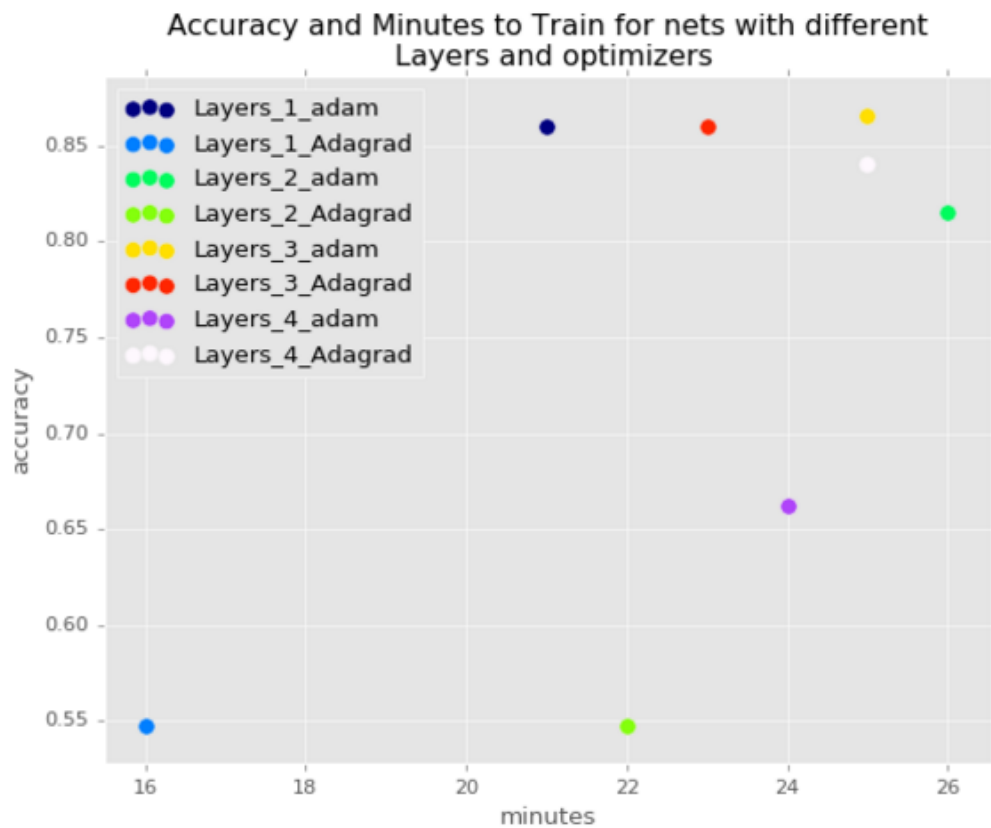
In the code side, `models.run_model()` is used to train the model as well as to validate it over the validation set and generate the learning curves. `models.build_model()` is used to build the model with the network architecture according to the parameters defined. Data generated in function `models.data_augmentation()` is used to train/test the model in batches of 32 images and 100 epochs, and to generate validation data to validate the trained model against new data not previously used during train/test. `models.learning_curves()` are used to graphically represent the process and get an overview of the accuracy and loss values by epochs.



*Illustration 11: Workflow representation. Data augmentation data is used to train/test and validate trained model and generate results and save trained model.*

## Refinement

Nets with 1 to 4 layers are tested in order to determine which configuration provides the best performance while minimizing computational resources. In the picture below it is demonstrate nets performance in terms of accuracy and minutes to train for the different layers and optimized used:



*Illustration 12: Summary results*

Net Architecture	Accuracy	Minutes to train
------------------	----------	------------------

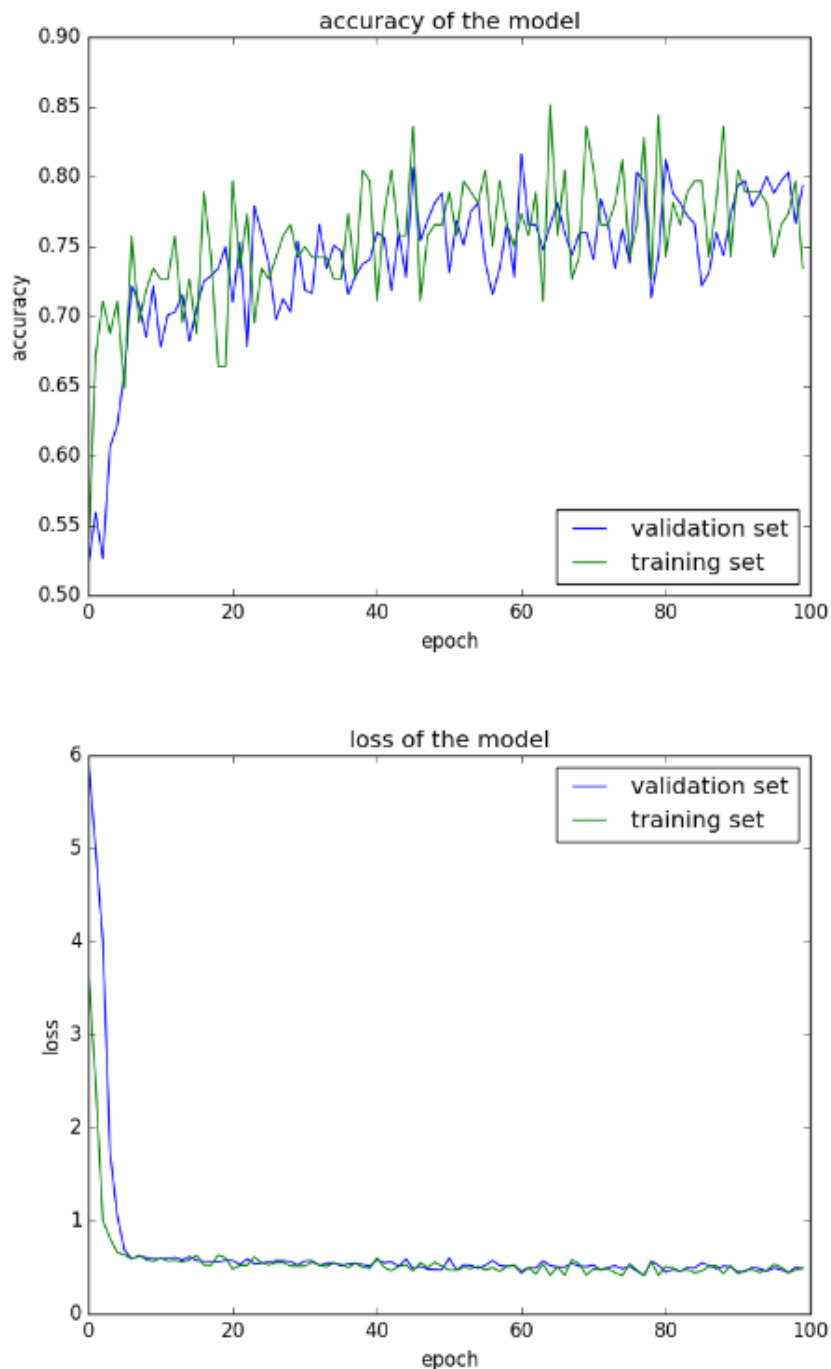
Layers_1_adam	0.86	21
Layers_1_Adagrad	0.55	16
Layers_2_adam	0.81	26
Layers_2_Adagrad	0.55	22
Layers_3_adam	0.87	25
Layers_3_Adagrad	0.86	23
Layers_4_adam	0.66	24
Layers_4_Adagrad	0.84	25

Neural Net with 1 layer and adam optimizer already meet the benchmark criteria, so no need of further refinement is required. Below are the accuracy and loss representation for Layer 1 and adam optimizer:

## IV. Results

### Model Evaluation and Validation

A validation set with a 10% of the dataset, not used during training/testing phase, is used to validate results. The final architecture selected, 1 layer and adam optimizer, reaches an accuracy over 80% which is in the range of the benchmark results. An accuracy of 80% means that model is correct in 80 out of 100 predictions made. Since the dataset is balanced (thanks to the data augmentation), accuracy is a perfectly valid metric in this scenario and no need to investigate alternatives like Precision & Recall or F-Scores is required.



*Illustration 13: Model results representation by epoch*

## Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- Are the final results found stronger than the benchmark result reported earlier?
- Have you thoroughly analyzed and discussed the final solution?
- Is the final solution significant enough to have solved the problem?

The result obtained with the model selected was higher than actually expected. In a more simplistic approach (just two classes), model is capable to reach state of the art accuracy performances even on the validation set (completely unseen data for the model). We can consider this proof of concept satisfactory as model reaches benchmark results.

## V. Conclusion

(approx. 1-2 pages)

### Free-Form Visualization

Following are represented some original pictures and how different filters represent them in the different convolutional layers. This gives us an idea of how the neural net decomposes the visual space.

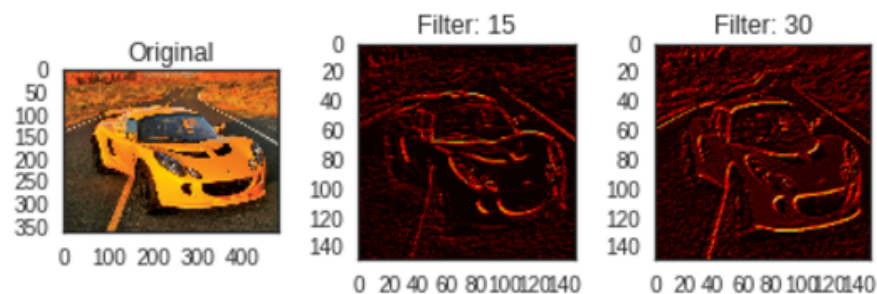


Image representation in Layer conv4

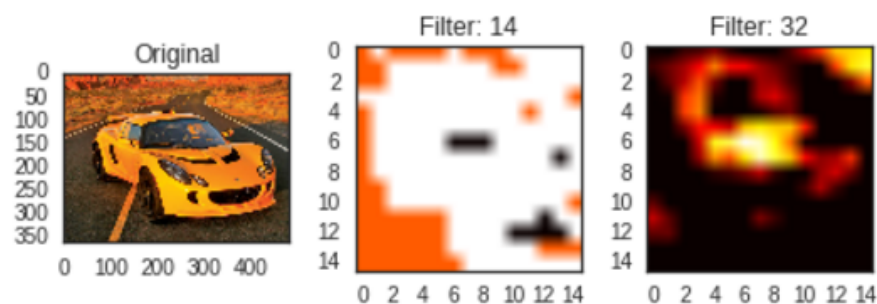
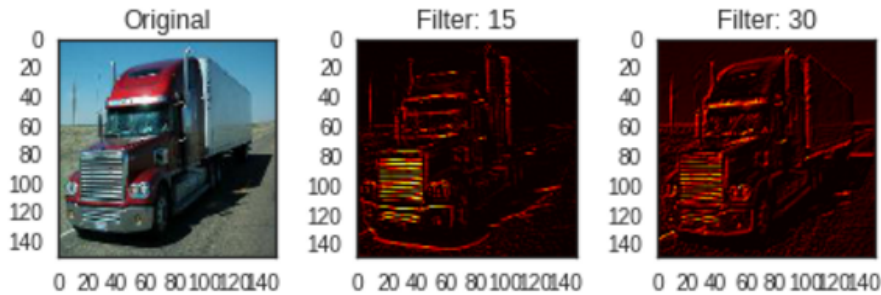
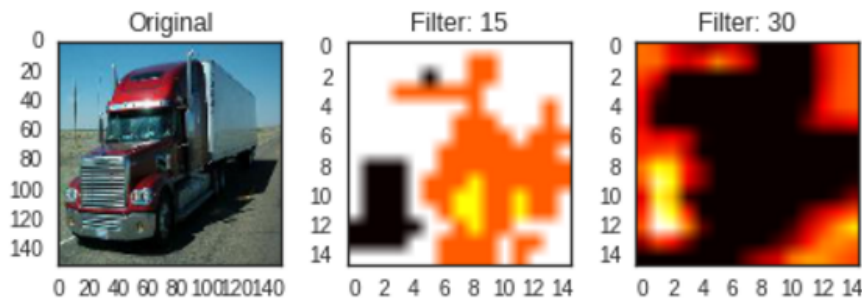


Illustration 14: Representation of the input image in different layers for different filters

### Image representation in Layer conv1



### Image representation in Layer conv4



*Illustration 15: Representation of a truck on different layers and filters.*

In both examples, for layer 1, different filters focus mainly on shapes and still images are recognizable, but in higher layers this not happens anymore and looks mostly noise. As mentioned by [@fchollet](#) on his expcetional [post](#): "Does it mean that convnets are bad tools? Of course not, they serve their purpose just fine. What it means is that we should refrain from our natural tendency to anthropomorphize them and believe that they "understand", say, the concept of dog, or the appearance of a magpie, just because they are able to classify these objects with high accuracy. They don't, at least not to any any extent that would make sense to us humans."

## Reflection

In this POC it is implemented a Deep Learning solution to automatic vehicle recognition. Image recognition used to be a difficult task historically, however for the last few years (thanks to augmented computational resources) there are efficient methods to approach these kinds of problems. Deep multi-layer neural networks are capable of building up a hierarchy of abstractions that makes it possible to identify complex inputs (i.e. images), and in this project this is the approach selected.

There were two major areas for the project. The first was data collection, the second was model building. Given that collected dataset is reduced, a critical part in this project is the use of the data augmentation utility from Keras to help to prevent overfitting and improve generalization.

After this, building the different models attempted is not particularly complex (thanks to Keras again!), and although there are a significant amount of parameters to experiment (such as the



type of activation functions, regularization methods, loss functions, error metrics, nodes in fully connected layers, etc.), it is started from good architectures that were published by [@fchollet](#), and build from there. It is amazing to see how efficient this method is, and how fast it is possible to set up an architecture that is performing well on the task.

Although the final method fits expectations for the problem, further testing with more validation data would be desired. The bottleneck here was the difficulty around data collection. Additional data could be used to warranty model does generalize well enough in largely different environments.

## Improvement

With regards to improvements, as already mentioned, gathering additional data would help in generalization. A further area in which to expand the project, is to expand it to a multiclass classification project, such that the model not only recognizes cars from trucks, but many other vehicles as well, such as vans, motorcycles, etc. Considering that, it would potentially be necessary to expand the model architecture by adding more layers and neurons to it, such that the model is expressive enough to accommodate the additional complexity.

## References:

[ref 01]: [Imagenet Classification with deep convolutional neural networks](#)

[ref 02]: [Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks](#)

[ref 03]: Ivakhnenko, A. G. and Lapa, V. G. (1965). Cybernetic Predicting Devices. CCM Information Corporation.

[ref 04]: [Wikipedia Deep Learning History](#)

[ref 05]: [Building powerful image classification models using very little data](#)

[ref 06]: [Convolutional Neural Network](#)

[ref 07]: [Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning](#)