

Effective solution of Visual Understanding competition from Kaggle

Definition

Project Overview

Kaggle is a platform for predictive modelling and analytics competitions in which companies and researchers post data and statisticians and data miners compete to produce the best models for predicting and describing the data. In this project I will present effective solution for one of the challenges from visual understanding field that I developed during this competition. The competition that I will solve in this project called "[Understanding the Amazon from Space](#)". It provides a data set that consists of images of Amazon taken from satellite. The goal is to build predictive model that will be able to label those images according to the objects that present on it. For example if river or road can be seen there then model should label it with appropriate tag.

Problem Statement

In this project I'll present solution for one of the Kaggle challenges. "[Understanding the Amazon from Space](#)" posted by Planet company. The input is a data set that contains RGB images of the Amazon taken from satellite. Some of them are labeled with one or several tags like river, road etc. We need to build predictive model that are capable to label new unseen images from the same dataset i.e. predict for each of 17 objects if it present on the image or not.

With recent advancement in Deep Learning HW and SW, development of algorithms for visual understanding became an easy task. However development of competitive algorithms for challenge purposes is hard problem. Finding optimal model for specific problem is hard task, since there is lots of different topology structures and hyper parameters that can be tuned. No single model can do better than ensemble of multiple models. The approach to solve this problem should combine several steps. Train multiple models and evaluate them on the validation set. Since we dealing with images as input the CNN models will fit best to this problem. Finally we need combine those using ensemble learning technics to achieve maximum performance.

Metrics

Metric that was chosen for this competition is F_β score. This is make sense since we dealing with very imbalanced classes in this dataset. In such cases accuracy metric may give misleading results. F_β calculated from combination of Precision and Recall metrics. Since we are solving multi-label classification task, averaging of individual F_β scores per class required to get global score. For this competition β was chosen to be equal 2. Bigger the β more the

Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

importance of the recall over the precision. For such imbalanced dataset like this it will motivate to select less frequent classes.

$$[1] F_{\beta} = (1 + \beta^2) \frac{p \cdot r}{p \cdot \beta^2 + r} \text{ where } p = \frac{T_p}{T_p + F_p}, r = \frac{T_p}{T_p + F_n}, \beta = 2$$

[1] <https://www.kaggle.com/c/planet-understanding-the-amazon-from-space#evaluation>

Analysis

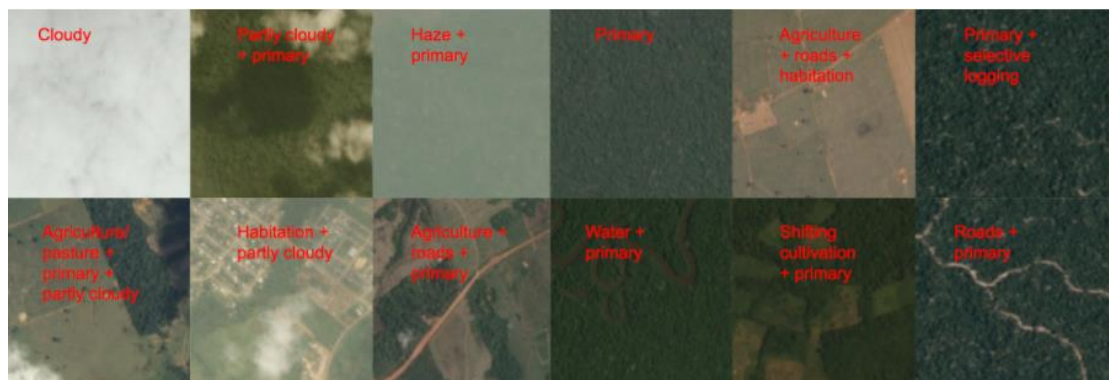
Data Exploration

Our dataset contains 40479 labeled training images and 61191 unlabeled test images. All images are RGB of size 256x256. We can deal with each image as 3d array with dimensions of [256, 256, 3].

Each image may have one or more labels up to 17.

Labels: ['slash_burn', 'clear', 'blooming', 'primary', 'cloudy', 'conventional_mine', 'water', 'haze', 'cultivation', 'partly_cloudy', 'artisanal_mine', 'habitation', 'bare_ground', 'blow_down', 'agriculture', 'road', 'selective_logging']

Fig. 1: Sample images with annotation from competition web page.



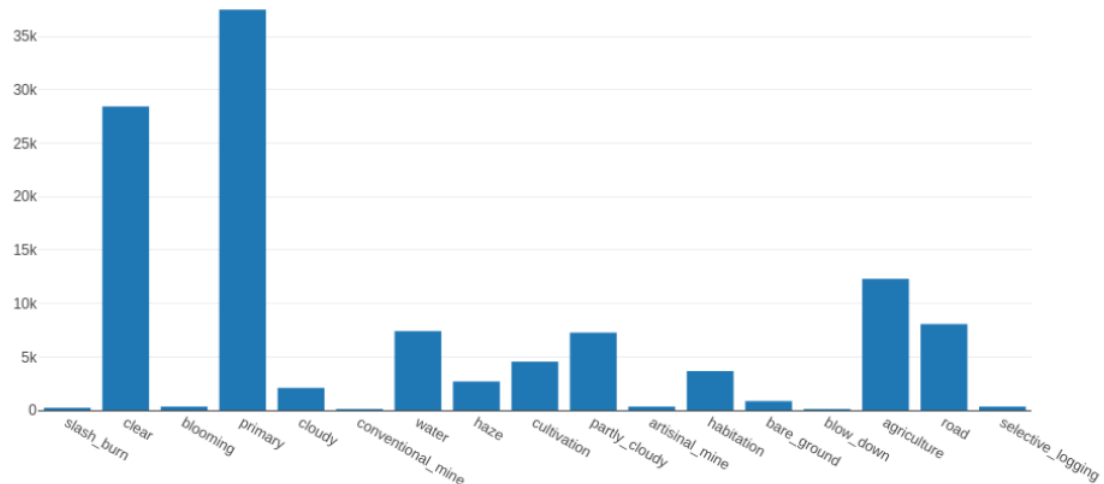
<https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data>

According to the organizers of the competition this data set was annotated by untrained personal and sometimes have mislabeling due to visual similarity of some tags. For example roads, rivers and selective logging may be confused with each other.

Exploratory Visualization

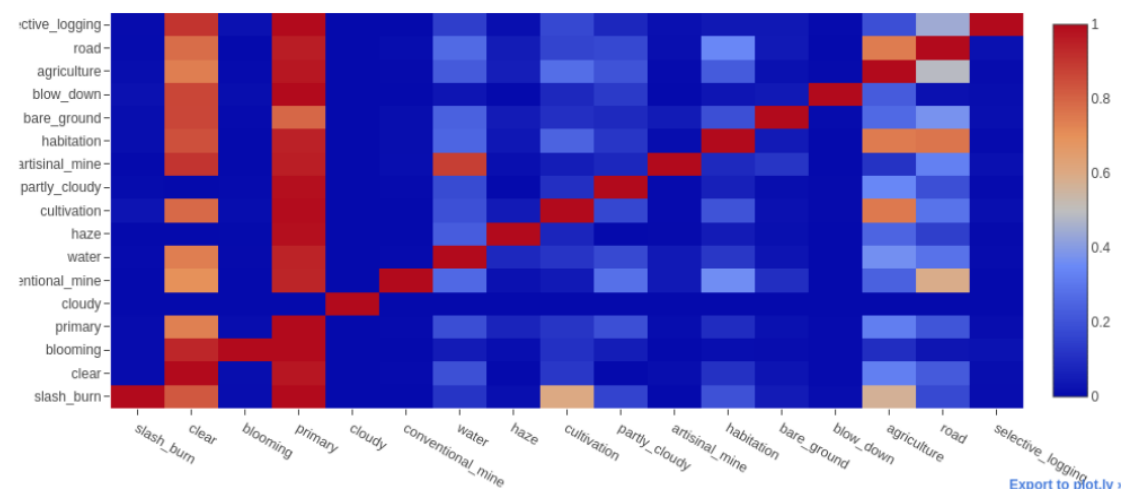
The plot below shows distribution of the labels among the training images. This is helpful to build reliable benchmark model.

Fig. 2a: Distribution of the labels across training images. Height of the bar shows amount of the images having certain label. Note that 'primary' is dominating with 35k images from total 40k training images.



First to mention is that data set have very imbalanced labels (fig. 2a). Approximately 80-90% of the images tagged with "primary" and "clear". Only few images labeled by "slash_burn", "blooming", "conventional_mine", "artisional_mine", "blow_down" and "selective_logging". Those labels will be very hard to learn due to scarcity, but on other hand their weight is small and should barely affect the total score.

Fig. 2b: Co-occurrence matrix of the labels. Useful to understand what tags more likely to appear together.



From fig. 2b again we can see that "primary" has biggest proportion among other labels and appear with all other labels except cloudy where the proportion of "primary" and "cloudy"

Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

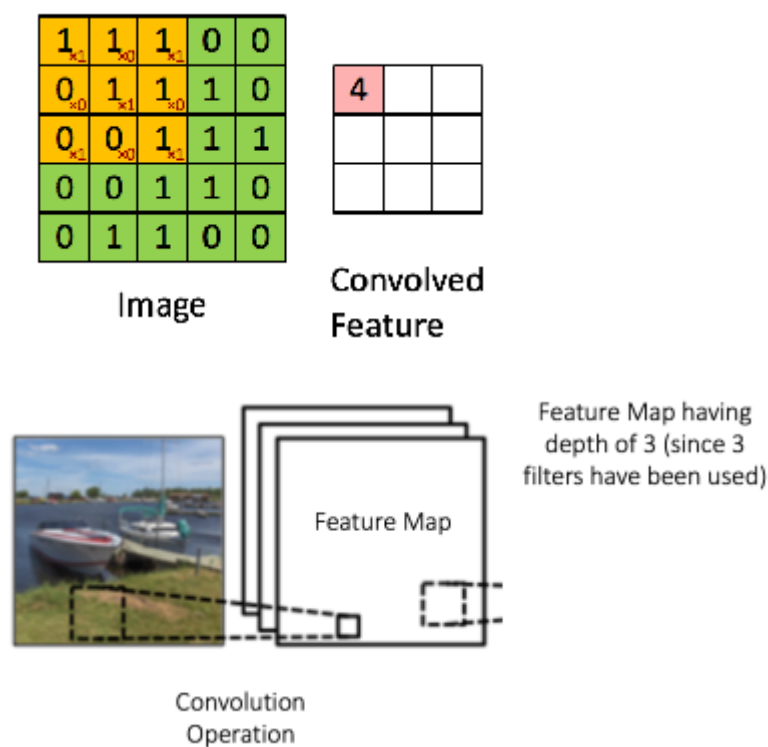
September 23, 2017

relatively low. Clear is also common but "clear" and "cloudy", "haze", or "partially cloudy" are mutually exclusive which is reasonable.

Algorithms and Techniques

In this project I used Convolutional Neural Networks to train a classifier. Convolutional Nets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a CNN is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. CNN works by sliding the filter over the image and computing the dot product of the input and the filter. The output called a "Feature Map" and it represents the features of the input.

Fig. 3a: Convolutional operation



Stacking those layers together and introducing non linearity in between allow to create more complex representations. Together these layers extract useful features from the images and reduce feature dimension.

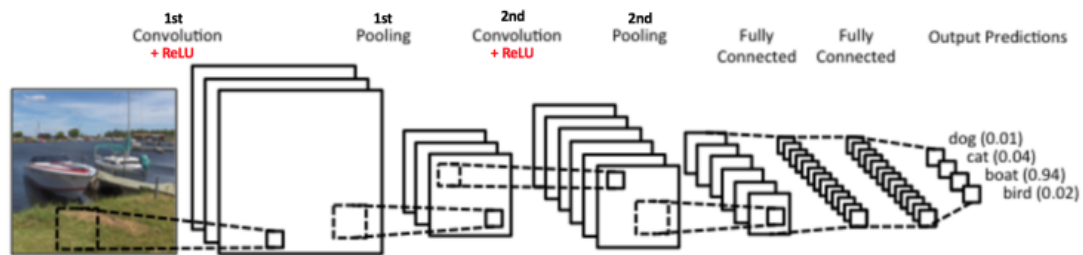
Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

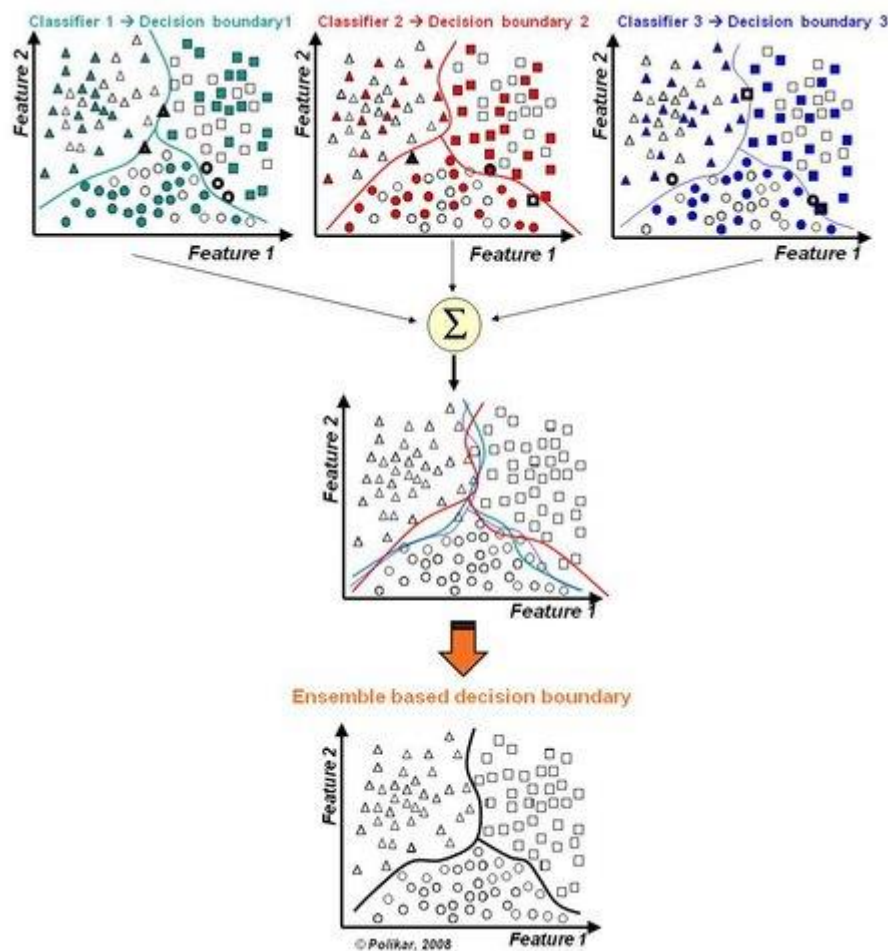
September 23, 2017

Fig. 3b: Convolutional network



To achieving competitive results it is not enough to have a single model. Combine multiple models can improve the results. Simplest way to combine models is averaging. Ensemble averaging is the process of creating multiple models and combining them to produce a desired output.

Fig. 3c: Averaging of three models results in better decision boundaries of classifier



Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

Benchmark

Let's evaluate performance of random model where tags selected randomly with respect to distribution of the labels in training set. Brief description of the steps.

- Split labeled data to training and validation set with proportions of 0.8 for training and 0.2 for validation.
- On the training set measure probabilities of labels by counting labels of specific class.
- For each image in validation and for each of 17 labels give 1 or 0 according to the probability distribution from previous step.
- Evaluate the performance of this model by calculating F_β score using the ground truth labels.

F_β score for this model is **0.56052**. Remind that in our case $\beta = 2$ so we calculate $F_2 = (1 + 2^2) \frac{p \cdot r}{p \cdot 2^2 + r} = 5 \frac{p \cdot r}{p \cdot 4 + r}$. According to precision recall trade off we know that if we increase precision recall decreases and vice versa. So with $\beta = 2$ we will prefer higher recall over higher precision.

- Define threshold between [0,1]
- Do exhaustive search over training data to find optimal threshold.
- For each image in validation data set label i to 1 if $p[i] > \text{threshold}$ otherwise set it to 0.
- Evaluate the performance of this model by calculating F_β score using the ground truth labels.

This model gives F_β of **0.69054** on validation set which is an improvement compared to previous. We can conclude that due to specifics of the metric, it is more rewarding to improve precision of the model than recall of relevant images.

Methodology

Data Preprocessing

Data loading: Data needs to be loaded to numpy array with dimensions [n, w, h, channels]. Where n is number of images in the data set.

Labels encoding: For each image we have variable number of labels up to 17. We will encode those labels in the vector of size 17. Where for each label vector v of specific image $v[i] = 1$ if this image has label i otherwise $v[i] = 0$. Dimension of the labels array is [n, 17].

Split for training/validation: training data set contains approximately 40k images. In addition to training we need a held out validation set so we need to split 40k images for training and validation parts. I choose to leave 20% of the data for validation, approximately 8k images and the rest of the images use for training.

Data augmentation: Data set provided by competition organizers consist of 40k training images and 60k test unlabeled images meaning we have only 40% of the original data set for

Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

training purposes. Taking into account that we need a held out data for validation leaves us with 32%. Supervise learning algorithms requires much more data to achieve good results and 32% definitely not enough. We can increase amount of training data by generating synthetic images from existing training images. This technic called data augmentation. In order to generate synthetic images I used 8 transformations. Combination of vertical and horizontal flips in addition to transpose of the images.

Implementation

I used Keras DL framework with Tensorflow backend to implement and train models. Training was done on Nvidia TitanX gpu and requires at least 64GB of RAM to load data.

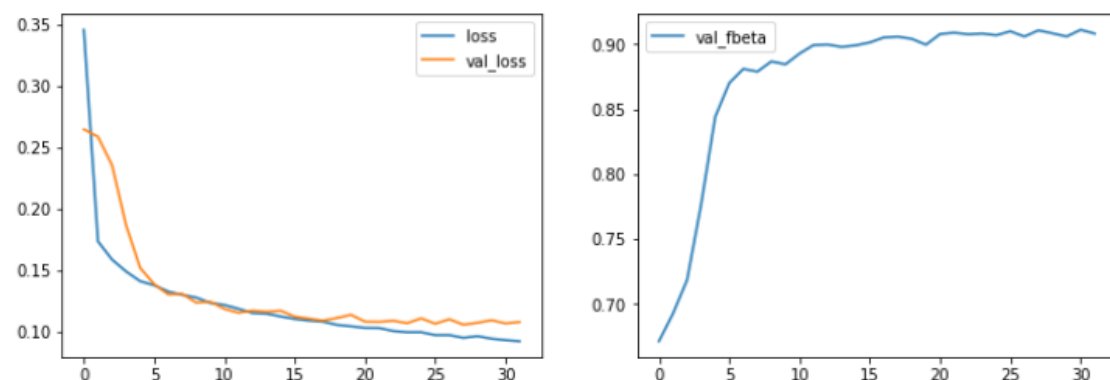
First step was to design simple sequential CNN model, train it and evaluate performance on validation set. This sequential model consists of 6 logical blocks fig. 4. Five blocks with same structure contains 2 convolutional layers, pooling layer and dropout. Those blocks used for feature extraction in order to learn spatial features from the images. Every subsequent block doubles the number of features and decreased the size of the feature map by 2 (max pool). Dropout used for regularization purposes in order to reduce overfitting on training set. Last block is non-linear classifier. Output of the classifier has 17 output neurons with sigmoid activation function (logistic function). In order to convert logistic levels to 1 hot I apply threshold of 0.2 on the outputs. Batch normalization layers added to the input and after dense layer improve convergence by normalizing inputs and classifier intermediate activations and slightly improves the score.

Fig. 4: Architecture of the sequential CNN used as initial topology.



Training this sequential CNN model achieves F_β of **0.91107** on validation set. This is huge improvement comparing to benchmark model score. Also loss curve is pretty smooth fig. 5a and well regularized despite that around epoch 20 start overfitting. F_β score improves gradually and reach maximum value around epoch 30.

Fig. 5a: Loss and F_β score of sequential CNN model.



Machine Learning Engineer Nanodegree

Capstone Project

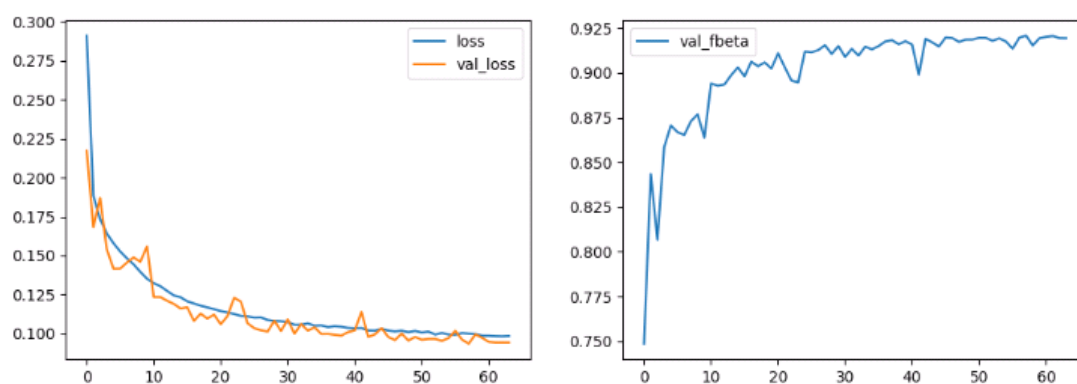
Yury Nahshan

September 23, 2017

Next experiment was to train simple sequential CNN from above on more data using data augmentation technics described in "Data Preprocessing" section. On more training samples this model achieved F_β score of **0.92088**. It feels like small improvement but top score on the public leaderboard for this completion was **0.93448**. Top 350 places competed in range between **0.92** and **0.93448**. So this simple model actually did pretty well and may end up in top 40%.

Interesting that adding data augmentation prevents overfitting. It can be seen from the validation loss that currently falls below training loss for whole duration of the training process. Both validation loss and score noisy probably due to small noise added to generated images.

Fig. 5b: Loss and F_β score of sequential CNN model with more training data.



My first approach to work with generated data was to generate and save all input images in numpy compressed format on the disk. Than load all this data and train regularly. This approach was very bad due to huge memory consumption around 128GB. Secondly every training epoch took 8 times longer. To resolve this I tried to use ImageDataGenerator API of Keras and generate augmented images on the fly. This resolves this issue with memory and longer epochs. Also it made training to be finer since the overall amount of data per epoch preserved only the overall amount of images for whole training process was increased 8 times on average.

Refinement

To make further improvement I used technic that called transfer learning. The best candidates for transfer learning donors are state of the art ImageNet models. ImageNet is a large visual database designed for use in visual object recognition research. Keras frameworks provides several pretrained ImageNet model and friendly API to retrain them on different data. I will show results of training two state of the art models Resnet50 and Inception v3 on competition data set.

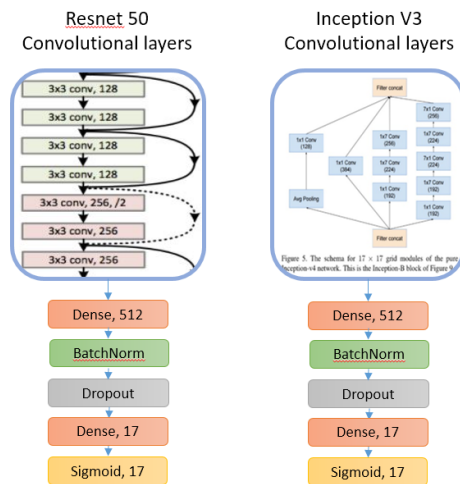
Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

Fig. 6: Transfer learning from Resnet50 and InceptionV3



In order to implement transfer learning Keras allows to load model and pre-trained weights without classifier on the top. Then just by adding same classifier from sequential CNN model on top of ImageNet model it can be re-trained on competition data set.

Table. 1: Comparison of the scores achieved by transfer learning from ImageNet models. Resnet50 outperforms InceptionV3 model.

model	score	loss
InceptionV3	0.923907	0.0891
Resnet50	0.927845	0.0862

If we compare scores for each class between models *Table 2*, we can see that different models do better on different classes. Resnet50 outperforms other in most of the classes, but even simple sequential model has best score on "partly_cloudy" class. So in order to benefit from different model we need to combine (ensemble) models.

Table. 2: Comparison of score per class for three different models. Best epoch corresponds to the epoch where this score was achieved.

Fβ score per class						
model	sequential		resnet50		inceptionV3	
class	score	best epoch	score	best epoch	score	best epoch
slash_burn	0	0	0.164835	38	0.08547	1
clear	0.979548	56	0.979981	21	0.97897	19
blooming	0.26764	54	0.326877	33	0.329815	17
primary	0.990891	61	0.991159	12	0.99119	21
cloudy	0.880414	55	0.886902	33	0.886386	25
conventional_mine	0.330189	63	0.702479	37	0.630631	24
water	0.79018	60	0.820829	35	0.815943	24
haze	0.773196	62	0.777075	25	0.776135	26

Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

cultivation	0.696043	53	0.700429	21	0.687465	21
partly_cloudy	0.947902	48	0.943558	9	0.944703	15
artificial_mine	0.871429	61	0.905172	38	0.901163	15
habitation	0.776473	47	0.795869	34	0.78794	24
bare_ground	0.329615	63	0.383023	38	0.409091	26
blow_down	0.078125	63	0.342466	29	0.227273	22
agriculture	0.89269	56	0.902229	35	0.896076	24
road	0.859657	57	0.8802	27	0.873444	24
selective_logging	0.46798	62	0.470297	10	0.478972	17

Additionally maximal score per class differs by number of epochs that required to train this model. As a consequence of this maximal total score doesn't mean that scores per class are maximized. What happen is that some "easy" classes take less epochs to learn and get to maximal score, where "hard" classes still not learned well. It takes more epochs for model to improve on those classes but "easy" classes starts overfitting. So to get better improvement I collect different checkpoints of each model and ensemble multiple checkpoints of different models.

There is a lot of different approaches and technics to ensemble models. Some kagglers even wrote a [blog](#) about it. I use greedy model selection algorithm which select best average of models on validation set. To reduce bias and improve generalization I use large validation set with augmented images exactly as for training.

Algorithm:

- 1: $M = [m_1, m_2, \dots, m_n]$ - bag of all models
- 2: split M to k bags M_1, M_2, \dots, M_k
- 3: for each M_i initialize $E_i = [m']$ where m' is best model in M_i
- 4: find m_j in M_i such that $F_\beta([E_i, m_j]) > F_\beta(E_i)$ and add m_j to E_i
- 5: repeat step 4 until can't improve E_i or no models left
- 6: final model = vote(average(E_i) for all E_i)

Results

Model Evaluation and Validation

Best score where achieve by ensemble of more than 100 models, where many of them was same model snapshot from different training epoch. All models where first validated against same validation set (Table 3) than submitted to the competition.

Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

Table. 3: Comparison of score for different models evaluated on original validation set of 8k images. After augmentation 32k images.

model	score	loss
Benchmark	0.69054	
Seq CNN	0.911074	0.1067
Seq CNN DA	0.920888	0.0935
InceptionV3	0.923907	0.0891
Resnet50	0.927845	0.0862
Ensemble	0.93348	

Justification




Test set for this competition consists of 60k images and separated into two parts public and private of 40k and 20k respectively. Evaluating on both validation and test sets gives approximately same F β score with small bias toward validation set (Table 4).

Table. 4: Evaluation of best ensemble of models on validation and test sets.

	F β score
validation set	0.93348
public test set	0.932
private test set	0.9307

Results show that ensemble methods proposed preserve generalization of the models to new unseen data and not overfitting on validation set despite that it built against validation set and not training set.

Solution presented in this project was submitted to the competition on Kaggle and took 37 place which is in top 4%. It can be found on competition's [leaderboard](#).

35	▲ 15	KILLER		0.93076	27	3mo
36	▲ 34	poteman		0.93075	5	3mo
37	▲ 29	Yura		0.93070	98	3mo

Conclusion

Free-Form Visualization

Fig. 7 presents images from the validation set with ground truth labels and predicted labels by resnet50.

Fig. 7.a: On the left image agriculture mistakenly predicted on this urban image, all other tags are correct. On the right image all tags are correct except water image which wasn't recognized.

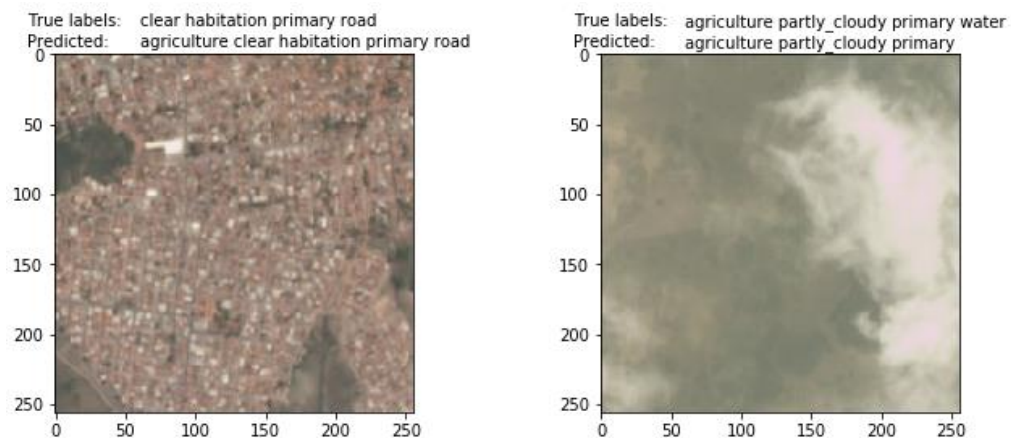
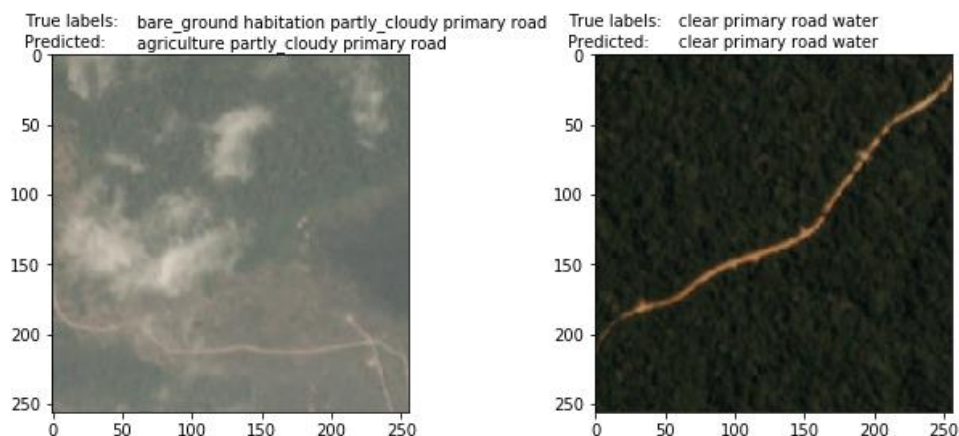


Fig. 7.b: On the left image bare_ground not recognized at all but we know that this is rare class. Left image also was tagged by agriculture instead of habitation but agriculture is more appropriate tag for this image so it's probably was mislabeled. Right image very clear and straight forward to understand. Both humans and resnet50 wasn't able to distinguish if it's road or water and tagged with both.



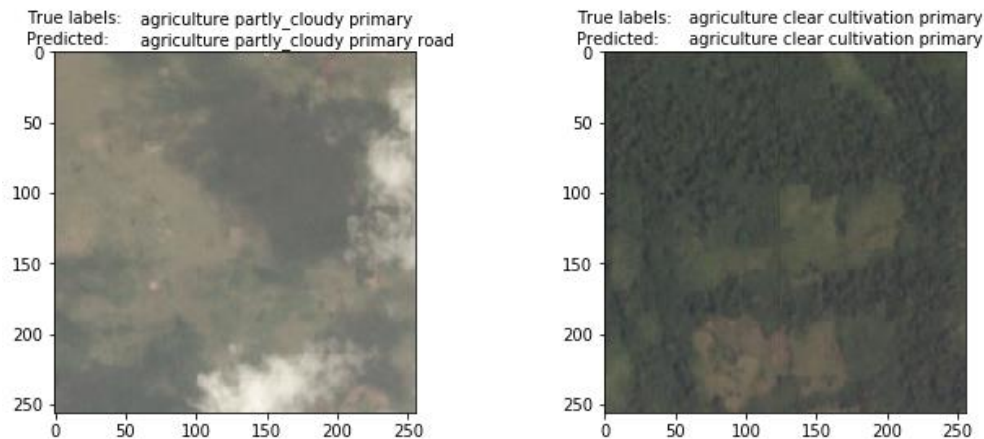
Machine Learning Engineer Nanodegree

Capstone Project

Yury Nahshan

September 23, 2017

Fig. 7.c: On the left image road mistakenly was predicted for some reason, all other tags are correct. Right image is simple and contains frequent well distinguishable tags, all tags was correctly predicted on it.



Reflection

General flow for this project can be summarized by following steps:

- Inspect the data
- Generate more images for training (Data Augmentation)
- Automate training process to train more models and save snapshots
- Train different models from more simple to complex
- Use transfer learning from state of the art models like ImageNet
- Use ensemble learning to achieve maximal possible score using those models

Some of the steps are specific to image classification but some are general. For instance ensemble learning is very wide and general domain of machine learning. It was not easy to select the approach that fit this project best. Another difficulty was to deal with a lot of data and big models where training time is meter of 10 hours. It required to created scripts to automate the training of the models and wisely select what models to try.

Improvement

To achieve better score it is possible to improve the steps described above.

Data augmentation may be further increased by implementing more reliable transformations like rotations of 45 degree. I didn't used this rotation since Keras library distorts the image when rotation is not 90 degree wise. With better implementation this transformation may be useful to generate more data and improve training.

Other ensemble technics instead the one I used can be evaluated and achieve better results. There is a vast amount of different technics to ensemble models [\[link\]](#). Doing more optimal ensemble can bring a lot of improvement.