A glance into competitive data science: the best practices for computer vision

How several techniques can dramatically improve the performance of your computer vision models¿

Pierre-Antoine Bannier

Jun 13, 2019 · 11 min read

Introduction

Since the Freesound Audio Tagging 2019 competition is coming to a close, I have decided to write an article about the best practices and techniques for building a robust computer vision model, that I¿ve learnt during my first research competition. Furthermore, it is a way for me to lay my ideas on paper in order to have a structured and clear overview of the techniques I¿ve used: the ones that worked well, as well as the ones that didn¿t yield a significant improvement in my test score.

As a student of the Fast.ai course, I¿m still at the very start of my learning journey, hence some remarks or techniques that could seem obvious for the most experienced of you.

As always, I strongly encourage you to correct me, to ask me precisions on a specific point if you don¿t get it or if it is unclear, and to add some remarks in the comment section!

The most critical lesson that I get from this competition is that building robust models is an empirical process. Competing helps develop an intuition for what might work or not.

To give you a bit of context, the competition is about tagging (classifying) sounds (cat, waterfalls, musical instruments¿). We were given two datasets: a curated one with clearly-audible sounds (4970 entries) and a noisy one with a sound and some noises in the background (20000-odd entries). Note that each data entry has multiple labels. Both datasets were labelled. The main challenge of the competition (identified by the vast majority of the participants) was to correctly use the noisy dataset to enhance the performance on the curated data.

We were evaluated with respect to a custom metric: the label-weighted label-ranking average precision also abbreviated lwlrap. To give a scale, the top 1% achieved a lwlrap score of 0.76, while a bronze medal (top 12% roughly) was awarded with a score of 0.695 or above. My final score during stage 1 was 0.686, which put me in the top 15%.

Fast Fourier Transforms and melspectrograms: how to feed sounds into a CNN?

Since the beginning of the article, you might find it weird that I speak about computer vision and sounds at the same time. Indeed, a sound is a time-based signal composed of multiple samples. At first, we could think that we can feed a sound into a CNN by applying 1D convolutions or into a LSTM-based model. However, both techniques did not yield convincing results, probably due to the loss of information incurred by feeding a raw signal into a model. Feeding a raw signal leads to a significant loss of information namely, frequency, phase, amplitude. After some experime

nts, both models did not yield a lwlrap over 0.5. Quite disappoi nting¿

As a rule of thumb for audio classification, we usually transfor m the sounds into a much more comprehensive 2D representation (a n image) that we feed into a CNN. This representation can be a M FCC, a Chroma, a CQT-Transform, but it is often a mel-spectrogra m.
Mel-spectogram of a signal plotted with Matplotlib

To understand how a mel-spectrogram is generated, we first need to understand what a Fourier transform does to our signal. The F ast Fourier Transform (a modified algorithm with a lower complex ity o(nlogn) than the original version o(n2)) converts a sound s ignal from its original time domain to a representation in the f requency domain.

Then we apply a series of transformations to finally get the mel -spectogram representation.

In order to perform this series of calculations, you can directl y use the librosa library that has various functions to perform audio feature analysis.

Additional readings:

- How to convert sounds into images in Python (a kernel by Dai sukelab): https://www.kaggle.com/daisukelab/creating-fat2019-pre processed-data
- A set of notebooks for audio features analysis: https://musi cinformationretrieval.com/
- An introduction to the librosa library in Python: https://to wardsdatascience.com/audio-classification-using-fastai-and-on-th e-fly-frequency-transforms-4dbe1b540f89

Mixup: a must-have to achieve state-of-the-art results in comput er vision

Simply put, Mixup is a data augmentation technique that creates new data entries from a linear relationship between two existing data entries. We can sum it up with this formula. ¿ are tensors that represent images, while ¿ are weights.

The same transformation is applied to the target:

It is no more than a weighted average of two-existing data sampl es. For instance, we could choose ¿ = 0.7. If ¿1 represents a do g and ¿2 a cat, ¿new will represent something between a dog and a cat, nearer from the dog than the cat. If you look at the imag e associated to ¿new, it may not make a lot of sense for you, bu t for the computer it clearly sees a dog, hence augmenting the d og image dataset.
Representation of a new data sample using Mixup (credits: Fast.a i)

As you can see it can act as a powerful data augmentation techni

que by creating new data from other existing pieces of data. It is particularly useful when you have to deal with class imbalance. It was the case during this competition (where some labels had very few occurrences, while others had a lot more). This single technique helped me break the 0.6 lwlrap barrier, from 0.57 to 0.65 on leaderboard (combined with other techniques listed below).

Additional readings:

Mixup original research paper: https://arxiv.org/abs/1710.09412
Fast.ai documentation on Mixup : https://docs.fast.ai/callbacks.mixup.html

Training Time Augmentation (TTA): scraping ranks in the leaderboard

Training Time Augmentation (TTA) is a form of data augmentation technique used during inference. Results from TTA are obtained by computing the weighted average of standard predictions and predictions made on your dataset after applying data augmentation techniques. Unsurprisingly, the equation looks very similar to the Mixup¿s:

By default, in fast.ai the beta coefficient is set at 0.4.

Note that in Fast.ai, TTA computes log-probabilities (hence the negative numbers). To pass from log-probabilities to ¿standard¿ probabilities (between 0 and 1), just compute the exponential of the probabilities.

The only issue that you can face is how long the model takes to compute the predictions. Especially if you have tight time constraints for the execution of your model, TTA might not be a good choice.

As for the competition, it made me jump (with Mixup) from 0.57 on leaderboard to 0.65.
Transfer, semi-supervised learning and pseudo labelling

Transfer learning consists in pre-training your model with a dataset and then training it with the dataset that is the closest to what you want to predict. It is a technique that allows a faster training and a steadier convergence to the loss optimum. It is now a standard in the industry, be it for computer vision or NLP with ULMFiT-based models and BERT.

For this competition, I chose to first pre-train my model on the noisy dataset to train my model on the curated dataset eventually. On my local machine, my lwlrap jumped from 0.81 to 0.83. Let ¿s recall that the competition did not allow pre-trained models.

The second technique I tried was semi-supervised learning. It is particularly useful if you have a lot of unlabeled data. It helps the model map the data distribution and allows a faster conve

rgence.

You better take one week to label additional data than tryin g to create a model that would perfectly take advantage of the u nlabeled data.

I chose one particular type of semi-supervised learning: pseudo-labelling the noisy data. In this competition, we had a large nu mber of noisy data. Rather than using the existing labels, I tra ined a model on curated data, then used this model to label my n oisy data. Eventually, by combining the curated data with the ne wly-labelled noisy data, I re-trained my model. Although, it did not improve significantly the performance due to too much diffe rence between the curated data distribution and the noisy one, i n an ensemble model, the performance increased.

I may conclude by stating that you better take one week to label additional data than trying to create a model that would perfec tly take advantage of the unlabeled data. Semi-supervised learni ng techniques are typically suited when you have large amount of spare data.

Additional reading:

A comprehensive explanation of pseudo labelling: https://www .analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervis ed-learning-technique/

K-Fold Cross Validation and bootstrap aggregating: achieving mor e stable results during inference

Since K-Fold CV is not particularly highlighted in the fast.ai c ourse, I haven¿t really used it although it is a standard in the industry. Let me quickly recall the definition of K-Fold valida tion: it is a technique whereby a dataset is divided into K-Fold , a model is trained on K-1 folds and validated on the remaining fold. Obviously, K models are generated since each model has a different validation set. This leads to more accurate and consis tent performance assessment.
A 5-fold cross-validation scheme

But beyond being a validation technique, K-Fold CV can be used w ithin an ensemble model (the fancy word for that is bootstrap ag gregating). This word hides pretty straightforward and simple te chniques where for instance you compute the average of all the p redictions on your K folds. This ensemble technique usually yiel ds better performance since the variance of the overall model is dramatically reduced.
Ensembling technique: the key to enter the top 5%

My previous paragraph introduced a key notion for competitive da ta science and broadly speaking for creating more accurate model s: the concept of ensembling. Simply put, ensembling is a techni que where several models are put together to gain stability in y our predictions and precision.

To achieve a highly-performant ensemble model, you need to check the correlation between all your models. The less correlated, the better. In order to do so, you compute the Pearson correlation coefficient:

Where ¿ is the standard deviation of the statistical series, cov (X,Y) the covariance of the statistical series X and Y.

By combining a simple baseline model with only curated data, a transfer learning model and a pseudo-labelling model, I was able to achieve my highest score on the leaderboard: 0.686.

To make it clearer, the statistical series X and Y are the predictions of your models. For each model that you create, you can compute the Pearson correlation coefficient between each other, and discard the models that are highly correlated.

In this competition, X and Y would be matrices that contain the probabilities of belonging to one of the 80 classes for every data sample.

As for the competition, the ensembling technique clearly gave me the opportunity to come close to a bronze medal. By combining a simple baseline model with only curated data, a transfer learning model and a pseudo-labelling model, I was able to achieve my highest score on the leaderboard: 0.686.

Additional reading:

A comprehensive guide for ensembling technique, from averaging to stacked generalization: https://mlwave.com/kaggle-ensembling-guide/

Where to go from here?

It is now time to adopt a critical point of view on my performance. Since I did not earn a medal, there are a lot of things to I could have improved. I am going to force myself to think about things that I could have done differently, or in addition to what I have already done.
Know your data

Probably my most critical error consisted in rushing too fast to build a model, instead of first trying to understand the data distribution and the kind of noise in the noisy dataset. It would have allowed me to better understand the difficulty of predicting a label and to read more relevant research papers to remove or attenuate the noise on the tracks.
Read research papers

This one is more of a reminder than an actual critic. I discovered how capital reading research papers was to apply techniques that lead to state-of-the-art results. With the democratization of AI thanks to MOOC such as fast.ai course (thanks again Jeremy Howard, Rachel Thomas and the fantastic team behind the project), more and more people have access to powerful machine learning

tools that bring them closer to state-of-the-art results. Readin g research papers and being able to implement some techniques de scribed is an invaluable perk since it differentiates you from t he competition. Few people are willing to take the time to read it and even fewer to implement a technique.

One of the main objectives that I set for myself is being comfor table enough with Fast.ai and its underlying framework PyTorch t o implement and integrate my own snippets of code to the library . For instance, I stumbled across this research paper https://ar xiv.org/abs/1905.02249 which seems to be quite efficient since i t is stated that MixMatch can reduce by a factor of 4 the error rate. However after multiple attempts, I am not yet able to impl ement it.

Using more data channels?

We¿ve talked previously about mel-spectograms and how they were created. But other audio representations can be created namely, Chroma, Mel Frequency Cepstral Coefficients (MFCC), Constant-Q T ransform to name but a few. All these representations emphasize on different aspects of an audio signal: one might explore the a mplitude while the other the phase or the frequency. I was not a ble to build an accurate enough model that would learn from the phase for instance. I strongly believe that it was a key of the competition since it adds more information to the mel-spectogram models.

More complex ensembling techniques: weighted average, stacked ge neralization?

While using an ensemble technique, I did not combine my models i n a complex way. I merely computed the mean of all my prediction s. I could have tried to fine-tune my model by choosing adequate weights, hence creating a weighted average for my predictions.

I could have even tried to use a neural network that takes as an input all the predictions from my different models and which tr ies to detect non-linear patterns between the different predicti ons. One challenge was data limitation. We only had 4970 data sa mples in the curated dataset, and removing 1000 data samples wou ld have been counterproductive, since 4970 was already a low num ber.

I did not try to train my model on the noisy dataset, since the distribution was too far from the testset¿s.

Conclusion

I¿m going to be short for the conclusion, since the article is a lready quite long (congrats and thanks if you are still here). T he most critical lesson that I get from this competition is that building robust models is an empirical process. You try, you fa il, you think, and you correct. You have to try a lot of things in order to become good at it. Competing helps develop an intuit ion for what might work and what might not.

I hope you liked this article. Again, don¿t hesitate to correct me if I am wrong on certain points or to share your thoughts on

the competition in the comment section.

And check my other blog post: https://medium.com/@PABTennnis/how-to-create-a-neural-network-from-scratch-in-python-math-code-fd874168e955

Kaggle: https://www.kaggle.com/rftexas