STFT Transformers for Bird Song Recognition

Jean-Francois Puget [1]

[1] *NVIDIA, 400 avenue Roumanille, BIOT, 06410, France*

Abstract

State of the art for audio classification often relies on the use of convolution neural networks on images made of (log mel) spectrograms. Recently, new image classification models, namely Vision Transformers (ViT0, have been proposed. ViT work by mapping non overlapping image patches to the input of a vanilla transformer model. While radically different from convolution neural networks their performance is almost as good. Inspired by this we have explored various ways to adapt ViT to spectrograms classification. The most effective way we found was to use time slices of spectrograms as the input patches. We call this model a STFT Transformer. This model has been evaluated and used during the BirdCLEF 2021 competition. This model alone would have got the 15th rank out of 816 competitors. When blended with some convolution networks it led to a 11th rank and a gold medal in that competition.

Keywords

Vision Transformer, Audio classification, Bird song recognition, CEUR-WS

1. Introduction

Bird song recognition is key for monitoring environment evolution and detect possible degradations. The use of machine learning to predict which species are present on audio recordings is a key enabler. A lot of progress has been possible by leveraging the extraordinary progress of computer vision machine learning models in the last decade. This progress is visible through BirdCLEF, a yearly competition series focusing on bird song recognition from audio recordings. Models that win one of these competitions are baselines for the next one. Progress is steady.

The 2021 edition [1][2], hosted on Kaggle, was no exception. Top solutions from previous year competition were used as baseline, eg [3][4]. Participants had to find novel techniques to beat previous year baseline. We present a novel technique we used, called STFT Transformer. It was inspired by recently image classifications models called vision transformers [5][6].

Using STFT Transformer is in line with what is now standard practice. Audio recordings are transformed into two dimensions images via the application of short term Fourier transform (STFT) on overlapping windows on the signal. The resulting image is called a spectrogram and has two dimensions: a frequency axis, and a time axis. The spectrogram can optionally be further transformed by log and mel transforms to get a non linear frequency axis. Figure 1 shows a 5 second waveform and its log mel spectrogram.

Once we have 2D images we can use all the machinery developed for image classification, including state of the art convolution neural networks (CNNs) like efficientnet. We can also use image augmentation techniques to make models more robust.

[1]CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
EMAIL: jpuget@nvidia.com
ORCID: 0000-0003-1978-0791

© 2020 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

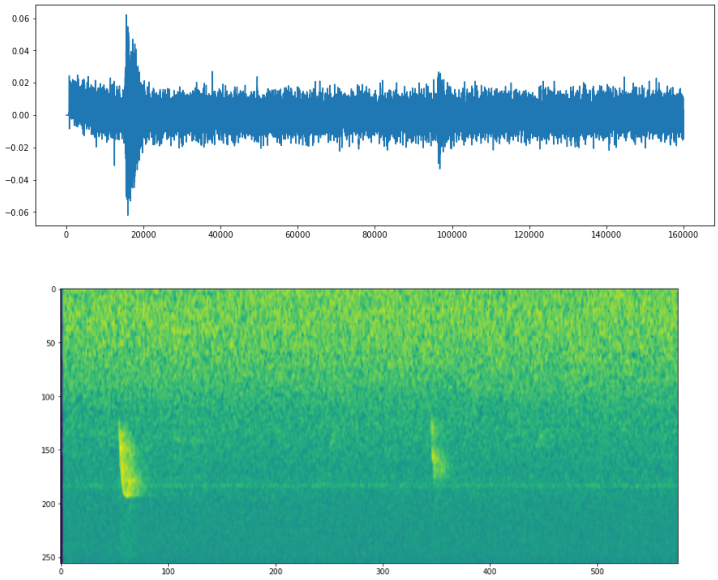

Figure 1: A waveform (top) and its log mel spectrogram (bottom)

Using CNNs on spectrograms must be done carefully though, because spectrograms are not ordinary images. The two axes of spectrograms (frequency and time) are not interchangeable. Some commonly used image augmentation techniques such as image rotation make no sense with spectrograms. A further issue is that CNNs are good at learning translation independent patterns. This is good for the time axis but is detrimental on the frequency axis as it corresponds to pitch change. We don't want models that ignore the pitch. STFT Transformer solves this problem entirely.

The contribution of this paper is four folds:

- A new STFT Transformer model working on short term Fourier transform sequences directly
- A way to reuse pretrained vision transformer model weights for STFT Transformer
- A robust training pipeline for STFT Transformer fine tuning
- Competitive results on the 2021 BirdCLEF competition

The paper is organized as follows. We describe STFT Transformer in section 2, and the 2021 BirdCLEF competition in section 3. We discuss how the model was trained on BirdCLEF data in section 4. We report competition results for STFT vs CNNs in section 5. We conclude and discuss possible improvements in section 6.

2. STFT Transformer

Using transformers to audio spectrograms is not straightforward. One of the issues is how to train them with audio data. Like any transformer model, this would require a lot of data. For comparison, ViT [5] has been trained on a 300M images dataset. The BirdCLEF competition training data is way too small for reliable training with about 64k recordings only.

One way to cope with small data is to use a pretrained model and fine tune it on the competition data. Unfortunately, there are no pretrained transformers for spectrograms. We decided to reuse pretrained weights from vision transformers pretrained on large image datasets. For this we needed to use spectrograms of the size expected by vision transformers. ViT expects an image of size 384x384 pixels. It then extracts non overlapping 16x16 patches arranged in a 24x24 grid.

Our first try was to create 384x384 spectrograms. Results were very disappointing, mostly because the time axis was too short, i.e. the number of STFTs was too small. We then realized that we could use a different grid shape while keeping the number of patches unchanged. We tried 12x48 patches (192x768 spectrogram), and 16x36 (256x576 spectrogram). Results were better because the spectrogram could include more STFTs, but still way below CNNs.

One reason for poorer performance was because ViT adds learnable position embeddings to its input patches. These embeddings capture the spatial relationship between patch locations. When we change the grid the position embeddings are no longer correct.

We implemented an interpolation from the 24x24 grid embeddings to the new grid size, either 12x48 or 16x36 grid. This interpolation was called after pretrained weights were loaded into the model and before training. As a result, the position embeddings captured the new spatial information accurately. This improved the model accuracy a bit, but most importantly, it made training convergence easier.

After revisiting the grid shape, we revisited patches shapes. One time slice of the 256x576 pixels spectrogram (one STFT) contains 256 pixels. And 256 pixels is precisely the number of pixels in a 16x16 patch. Therefore, we decided to feed a 1x576 grid of 256x1 patches to ViT. The main consequence of this choice is that patches encompass all the frequency axis. There is no longer a need to worry about their frequency position. And there cannot be any pitch invariance in the model anymore.

Still ViT expects a 384x384 image. We therefore had to reshape the 256x576 spectrogram into a 24x24x16x16 image. Figure 2 shows how the spectrogram from Figure 1 looks like after the reshaping.

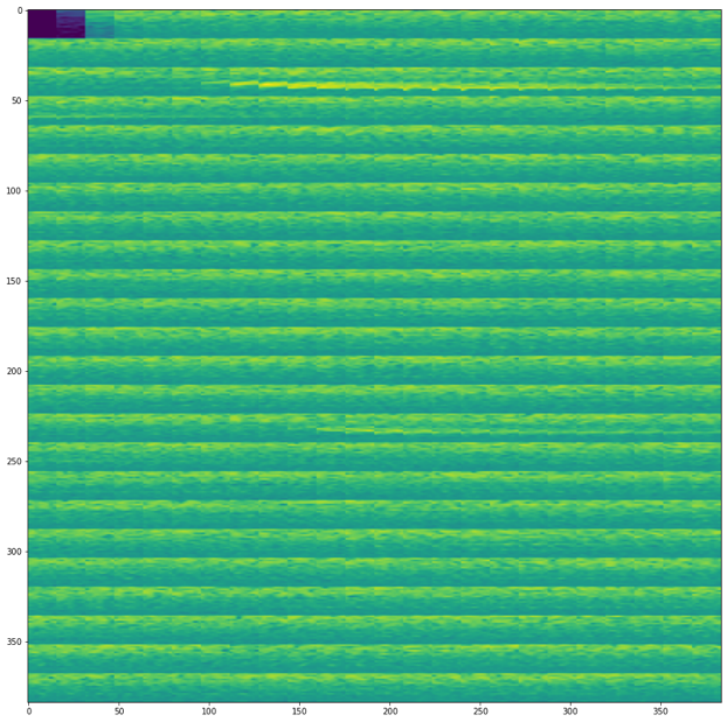

Figure 2: example of an input to STFT Transformer

We tried various ways to adapt the existing position embeddings to a one dimensional grid but the best was to just leave them as is. During fine tuning the model adjusted them to the one-dimensional sequence nature of the input.

We call the resulting model a STFT Transformer given it takes as input a STFT sequence. This model has been evaluated and used during the BirdCLEF competition, see below.

3. BirdCLEF 2021 competition

The competition was hosted on Kaggle [6], the largest machine learning competition worldwide, now owned by Google. Training data for this year competition came in two flavors:

- 63874 recordings from Xeno Canto [7]. Each recording comes with a primary label, i.e. the name of a bird species heard in the recording. It also has secondary labels, which are other species heard in the recording, but maybe less often and not as loud as the primary label. Duration of these recordings ranges from 5 seconds to minutes. Location, time of recording, and many other meta data are available for each of these recordings.

- 20 soundscapes of 10 minutes duration. Each 5 second clip of train soundscape comes with a list of bird species heard during the 5 seconds, or nocall if there is none.

Test data is hidden from contestants. It is made of about 80 soundscapes similar to the train soundscapes. The labels for test data are also hidden. There were 397 bird species present in training data. Competition hosts said that all species present in test data also appeared in training data. Test data was split between a public part and a private part that were kept hidden.

Contestants had to provide a script or a jupyter notebook on Kaggle [6] that reads test data, and that outputs a file containing predictions for each 5 second clip of each test soundscape, without being able to access any internet resource. Hiding the test data is a great way to test generalization of machine learning models to new data. Disabling internet access is to make sure no one can leak test data.

Contestants could train machine learning models as they see fit. Then they usually created a Kaggle dataset containing their trained model weights. This Kaggle dataset was attached to the test notebook. Code in the notebook could then load these weights from the attached dataset and use them. This way it was possible to use pretrained models on hidden test data.

Once contestant code was submitted it was run with the hidden test data, and two scores were computed. but the score obtained on the public test part was reported to the contestant. Contestant best public score was displayed in a public leaderboard. It was then possible to see the performance of all contestants in real time. This public leaderboard is a very effective gamification tool. It is very addictive, and most of the contestants checked it on a daily basis during the competition, if not often.

At the end of the competition scores computed on the private test part are disclosed in a private leaderboard. Ranking on the private leaderboard is the official result of the competition. Top 11 ranks were awarded a gold medal. Top 3 ranks also won a prize money.

4. Training models

Even with pretrained weights, fine tuning transformers is tricky. Pipeline designed for training convolution networks fail miserably with exploding loss functions, or disappointing model performance. We devised a training pipeline that enabled robust training of good STFT transformers.

4.1. Data preprocessing

We trained the model on the xeno canto recordings. It was clear that training on random 5 second clips had a drawback: some of the clips may not contain the target bird. We then used a simple hypothesis: clips were stripped, i.e. periods without a song at the beginning or at the end were removed for the sake of reducing storage needs. We therefore trained on first 5 seconds or last 5 seconds of clips, assuming these would contain the primary label bird. We preprocessed all Xeno Canto recordings and saved their first 8 seconds and their last 8 seconds as Numpy files. This speeded up data loading significantly given we did not need to load full recordings.

In order to improve model confidence when predicting no call, we added to our training data the nocall clips from freefield1010 dataset [8]. We added a 498th class to represent the nocall. As a result our model could predict both one or more birds, or a nocall.

Train soundscapes could contain several bird species in a 5 second clip. To model that out of mono bird clips from Xeno Canto we implemented a mixup variant were up to 3 training clips could be merged. To model the presence of several bird species we define the target for the merged clip as the maximum of the targets of each merged clip. The presence of nocall clips from freefield1010 added background noise when they were used via mixup.

We used librosa [9] to compute spectrograms. We used 2048 n_fft, 250 n_mels, and the default windows length. We computed the hop length as a function of the desired spectrogram length: a shorter hop length yields a longer spectrogram. The number of mels defines the height of the spectrogram. It

was then resized to the desired height. We contemplated moving spectrogram computation to GPU but we saw that the GPU was fully busy on Kaggle with model prediction.

We implemented pitch variation and speed variations by modifying the sampling rate and the hop length for the STFT computation routine from librosa. A higher sampling rate than the actual sampling rate moved the pitch upward. A shorter hop length amounted to using to a slowed down recording. Clip length was adjusted to yield the same spectrogram length as when no data transformation was used. It is why we stored clips longer than 5 seconds. It enables using long enough clips when hop length is increased. We also added gaussian noise used in previous year solution [3].

When predicting on soundscapes (train or test) we disabled all data augmentations. Soundscape was fed 5 second per 5 second to the model.

4.2. Loss function and secondary labels

Contestant submissions were evaluated by comparing the model predictions to the test ground truth. For each 5 second clip a F1 score between the predictions and the ground truth is computed. Then these F1 scores are averaged.

F1 score cannot be used directly as a loss function for training models because it is not differentiable. We simply used binary cross entropy for a 498 multilabel problem. For each training clip we defined a 498 target, with a 1 for the primary label or a nocall.

Primary labels were noisy, because there is no warranty that the primary label does appear in the first 5 seconds and in the last 5 seconds of every Xeno Canto recording. But they must happen often enough given the good result of our models. We hypothesize that the small proportion of cases where they do not appear in these clips acts as a regularization, a bit like label smoothing.

Secondary labels were even noisier than primary labels. And there, the clipping argument does not apply. Secondary labels could occur really anywhere in training clips. To cope with that we masked the loss for secondary labels as we didn't want to force the model to learn a presence or an absence when we don't know. We therefore defined a secondary mask that nullifies the BCE loss for secondary labels. For instance, assuming only 3 bird species b0, b1, and b2, and a clip with primary label b0 and secondary label b1, then these two target values are possible:

[1, 0, 0]

[1, 1, 0]

The secondary mask is therefore:

[1, 0, 1]

The loss for b1 is masked and ignored. For merged clips in mixup, a target is masked if it it not one of the primary labels and if it is one of the secondary labels.

4.3. Cross validation

We used two ways to evaluate our model strength. The first one, used by many contestants, is to use the train soundscape as holdout dataset. Models are trained on Xeno Canto clips, then used to make predictions on every 5 second clips of every train soundscape. Then F1 score is computed as for the test data.

The test data had 54% of no calls as witnessed by a nocall submission score. In order to adjust for the difference in the proportion of nocall clips in training soundscapes and test soundscapes we scaled the computation of validation score. We compute F1score on no call rows separately from F1 score on bird call rows, then compute final score with

$$score\_all = 0.54 * score\_nocall + (1 - 0.54) * score\_birds \quad (1)$$

Our validation score was very close to the submission score, except for some models from time to time. In order to detect we used 5 folds cross validation on xeno canto clips. We adjusted the score with equation (1) as well.

We then defined our local score as the minimum of the train soundscape score and the xeno canto cross validation score. This revised local score was very well correlated with leaderboard score.

4.4. Training transformers

We started training our modified vision transformers using the same pipeline as our CNNs, namely 60 epochs with Adam optimizer with weight decay and a one cycle lr scheduler. Given the class imbalance we used a posweight of 10 in BCE loss as well as a class weight that was proportional to the inverse of each bird species frequency. This worked very well in past competitions for CNNs. It worked very well here too because our CNN stayed on top of the public leaderboard during all the competition till the last week.

However, it was impossible to train transformers with it. Either loss was exploding, or the models did not train much. We hypothesized that it was because loss landscape was not smooth at all for vision transformers, with very narrow local minima. This hunch has been confirmed since then in a paper published after the competition deadline [10].

To make transformer learn we hypothesized that keeping gradient as small as possible would enable reaching very narrow loss minima. We tried gradient clipping but it was too crude. We finally ended up with using a combination of amp autoscale, no class weights, and a small learning rate of 1e-4. In order to cope with class imbalance we sampled clips according to a weight that depended on its primary label frequency. The less frequent, the more often the clip would be selected.

We tried several variants of ViT and only kept the original ViT [5] and its DeiT variant [11], mostly because they were the ones supporting 16x16 patches on images larger than 224x224. We used timm [12] implementation of vision transformers and other models.

5. Results

Our models were trained on 5 second clips. They may miss longer time span patterns as well as species co-occurrences. We could have addressed this by using a sound event detection model [13] with our models as backbone. Instead, we decided to postprocess our soundscape prediction logits using 3 thresholds tuned on train soundscapes, *pres_thr*, *incr_thr* and *occur_incr*:

1. A key observation was made in [3] and [4]: if a bird is heard in one of the 5 second clips of a soundscape then it is more likely to be heard in others. To capture this correlation, we computed the maximum prediction for each species across all 5 second clips. For species where this maximum is greater than *pres_thr* we predicted this species presence as soon as a 5 second clip prediction is above *incr_thr*.
2. If two bird species are known to co exist then hearing one makes the other one more likely. For this we computed a co occurrence matrix using primary and secondary labels. If a species was detected then logits for all co occurring species were increased by *occur_incr*.

Typical values for *pres_thr*, *incr_thr* and *occur_inc* are: 2.1, -0.2, 0.9.

We report results for 5 models, STFT Transformer, two vision transformers using 16x16 patches, ViT and DeiT, and two variants of our best CNN in Table 1. The two CNNs uses the same EfficientNet b3 backbone, but with different input sizes. For each model we report the validation score on training data, the public leaderboard score and the private leaderboard score.

Table 1
Model performance

Backbone	Image size	Val score	Public LB	Private LB
Efficientnet b3	300x600	0.7855	0.7906	0.6554
Efficientnet b3	200x600	0.7853	0.7965	0.6630
16x16 ViT	192x768	0.7645	0.7445	0.6317
16x16 Deit	192x768	0.7631	0.7580	0.6397
STFT DeiT	256x576	0.7755	0.7569	**0.6667**

We see that DeiT outperformed ViT when applied to spectrograms with 16x16 patches. Therefore, we did not submit a ViT version of STFT Transformer and only used the DeiT version. STFT Transformer looked weaker on the public leaderboard than the CNNs, but performed slightly better on the private leaderboard. We also see that STFT Transformer is significantly better than DeiT using 16x16 patches. We believe it is because the former removes the pitch translation issue that the latter faces.

STFT private score is 15th on the private leaderboard. Given most top teams used an ensemble of model we think STFT Transformer alone is in the top 10 models of the competition. When blended with the two CNNs in the table it led to the 11th rank and a gold medal.

6. Conclusion and future work

State of the art models for audio classification often rely on convolution networks applied to spectrograms. We presented a new model for audio clip classification, called STFT Transformer. This model is inspired by recently introduced Vision Transformers. Main difference is that our model takes as input the sequence of short-term Fourier transforms of the audio signal, instead of using 16x16 patches from spectrograms. We described how to reuse weights of vision transformers pretrained on large image datasets. We also described how the model was trained and used in the 2021 BirdCLEF competition. The model seems competitive with top convolution networks. We claim it is because STFT Transformer does not learn pitch invariant patterns.

Our model can certainly be improved in many ways. The most obvious is to use it as a backbone in a SED model [13] in order to capture long range patterns and species co-occurrences. We also would like to explore pretraining on alarger audio datasets instead of reusing models pretrained on image datasets. Last, but not least, we would like to use the new SAM optimizer [10] to train our STFT Transformers.

Our code is available at https://github.com/jfpuget/STFT_Transformer

7. Acknowledgement

This work was fully funded by NVIDIA. The author is part of a NVIDIA team whose mission is to compete on Kaggle and other platforms to demonstrate the effectiveness of NVIDIA hardware and software stack. We would like to thank the organizers and Kaggle for hosting a flawless competition. Data was clean, metric was well defined, and hosts provided great introductory notebooks.

8. References

[1] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, R. Ruiz DeCastañeda, I. Bolon, H. Glotin, R. Planqué, W.-P. Vellinga, A. Dorso, H. Klinck, T. Denton, I. Eggel, P. Bonnet, H. Müller, Overview of LifeCLEF 2021: a System-oriented Evaluation of Automated Species Identification and Species Distribution Prediction, in: Proceedings of the Twelfth International Conference of the CLEF Association (CLEF 2021), 2021.

[2] S. Kahl, T. Denton, H. Klinck, H. Glotin, H. Goëau, W.-P. Vellinga, R. Planqué, A. Joly, Overview of BirdCLEF 2021: Bird call identification in soundscape recordings, in: Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, 2021.

[3] R Wong, 1st Place Solution, 2020. URL: https://www.kaggle.com/c/birdsong-recognition/discussion/183208

[4] V.Kramarenko, 2nd place solution, 2020. URL: https://www.kaggle.com/c/birdsong-recognition/discussion/183269.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. Arxiv, 2010.11929, 2020.

[6] Kaggle. URL: https://www.kaggle.com
[7] Xeno canto. Sharing bird sounds from around the world. URL: https://www.xeno-canto.org/
[8] D. Stowell, Freefield1010, 2013. URL: https://archive.org/details/freefield1010
[9] McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, et al. librosa: Audio and music signal analysis in python. In: Proceedings of the 14th python in science conference. 2015.
[10] X. Chen, CJ. Hsieh, B. Gong, When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations arXiv:2106.01548, 2021.
[11] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablay-rolles, and H. Jegou. Training data-efficient image trans-ormers & distillation through attention. arXiv:2012.12877, 2020.
[12] R Whitman, timm, 2021. URL: https://github.com/rwightman/pytorch-image-models
[13] Q. Kong, Y. Cao, Y. Wang, W. Wang, M. Plumble, PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition, arXiv:1912.10211, 2020.