9th place solution: smaller and faster
by Iafossin Freesound Audio Tagging 2019 8 months ago

First, I would like to congratulate all winers and participants
of this competition and thank kaggle and the organizers for post
ing this interesting challenge and providing resources for parti
cipating in it. Also, I want to express my highest gratitude and
appreciation to my teammates: @suicaokhoailang, @theoviel, and
@yanglehan for working with me on this challenge. It was the fir
st time when I worked with audio related task, and it was a mont
h of quite intense learning. I would like to give special thanks
to @theoviel for working quite hard with me on optimization of
our models and prepare the final submissions.

The code used for training our final models is available now at
this link (private LB 0.72171 score, top 42, with a single model
(5 CV folds) trained for 7 hours). Further details of our solut
ion can be found in our DCASE2019 technical report.

The key points of our solutions are the following (see details b
elow): (1) Use of small and fast models with optimized architect
ure, (2) Use of 64 mels instead of 128 (sometimes less gives mor
e) with 4s duration, (3) Use data augmentation and noisy data fo
r pretraining.

Stage 1: we started the competition, like many participants, wit
h experimenting with common computer vision models. The input si
ze of spectrograms was 256x512 pixels, with upscaling the input
image along first dimension by a factor of 2. With this setup th
e best performance was demonstrated by DenseNet models: they out
performed the baseline model published in this kernel and succes
sfully used in the competition a year ago, also Dnet121 was fast
er. With using pretraining on full noisy set, spectral augmentat
ion, and MixUp, CV could reach ~0.85+, and public score for a si
ngle fold, 4 folds, and ensemble of 4 models are 0.67-0.68, ~0.7
0, 0.717, respectively. Despite these models are not used for ou
r submission, these experiments have provided important insights
for the next stage.

Stage 2:
Use of noisy data: It is the main point of this competition that
organizers wanted us to focus on (no external data, no pretrain
ed models, no test data use policies). We have used 2 strategies
: (1) pretraining on full noisy data and (2) pretraining on a mi
xture of the curated data with most confidently labeled noisy da
ta. In both cases the pretraining is followed by fine tuning on
curated only data. The most confident labels are identified base
d on a model trained on curated data, and further details can be
provided by @theoviel. For our best setup we have the following
values of CV (in stage 2 we use 5 fold scheme): training on cur
ated data only - 0.858, pretraining on full noisy data - 0.866,
curated data + 15k best noisy data examples - 0.865, curated dat
a + 5k best noisy data examples - 0.872. We have utilized all 3
strategies of noisy data use to create a variety in our ensemble
.

Preprocessing: According to our experiments, big models, like Dn
et121, work better on 128 mels and even higher image resolution,
 while the default model reaches the best performance for 64 mel
s. This setup also decreases training time and improves converge
nce speed. 32 mels also could be considered, but the performence
 drops to 0.856 for our best setup. Use of 4s intervals instead
of traditional 2s has gave also a considerable boost. The input
image size for the model is 64x256x1. We tried both normalizatio
n of data based on image and global train set statistics, and th
e results were similar. Though, our best CV is reached for globa
l normalization. In final models we used both strategies to crea
se a diversity. We also tried to experiment with the fft window
size but did not see a significant difference and stayed with 19
20. One thing to try we didn't have time for is using different
window sizes mels as channels of the produced image. In particul
ar, this paper shows that some classes prefer longer while other
 shorter window size. The preprocessing pipeline is similar to o
ne described in this kernel.

Model architecture: At stage 2 we used the model from this kerne
l as a starting point. The performance of this base model for ou
r best setup is 0.855 CV. Based on our prior positive experience
 with DensNet, we added dense connections inside convolution blo
cks and concatenate pooling that boosted the performance to 0.86
8 in our best experiments (model M1):

```
class ConvBlock(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=3,
 pool=True):
        super().__init__()

        padding = kernel_size // 2
        self.pool = pool

        self.conv1 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=ker
nel_size,
                stride=1, padding=padding),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(out_channels + in_channels, out_channels,
                kernel_size=kernel_size, stride=1, padding=paddi
ng),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
        )

    def forward(self, x): # x.shape = [batch_size, in_channels,
a, b]
        x1 = self.conv1(x)
        x = self.conv2(torch.cat([x, x1],1))
        if(self.pool): x = F.avg_pool2d(x, 2)
        return x   # x.shape = [batch_size, out_channels, a//2,
b//2]
```

The increase of the number of convolution blocks from 4 to 5 gav
e only 0.865 CV. Use of a pyramidal pooling for 2,3 and 4-th con
v blocks (M2) gave slightly worse result than M1. Finally, our u
ltimate setup (M3) consists of 5 conv blocks with pyramidal pool
ing reached 0.872 CV. DenseNet121 in the same pipeline reached o
nly 0.836 (DenseNet121 requires higher image resolution, 256x512
, to reach 0.85+ CV). From the experiments, it looks for audio i
t is important to have nonlinear operations before size reductio
n by pooling, though we did not check it in details. We used M1,
 M2, and M3 to create variability in our ensemble. Because of th
e submission limit we checked performance of only a few models i
n public LB, with the best single model score 0.715.

This plot illustrates architecture of M3 model:

Here, all tested models are summarized:

And here the model performance for our best setup:

Data augmentation:
The main thing for working with audio data is MixUp. In contrast
 to images of real objects, sounds are transparent: they do not
overshadow each other. Therefore, MixUp is so efficient for audi
o and gives 0.01-0.015 CV boost. At the stage 1 the best results
 were achieved for alpha MixUp parameter equal to 1.0, while at
the stage 2 we used 0.4. Spectral augmentation (with 2 masks for
 frequency and time with the coverage range between zero and 0.1
5 and 0.3, respectively) gave about 0.005 CV boost. We did not u
se stretching the spectrograms in time domain because it gave lo
wer model performance. In several model we also used Multisample
 Dropout (other models were trained without dropout); though, it
 decreased CV by ~0.002. We did not apply horizontal flip since
it decreased CV and also is not natural: I do not think that peo
ple would be able to recognize sounds played from the back. It i
s the same as training ImageNet with use of vertical flip.

Training: At the pretraining stage we used one cycle of cosine a
nnealing with warm up. The maximum lr is 0.001, and the number o
f epochs is ranged between 50 and 70 for different setups. At th
e stage of fine tuning we applied ReduceLROnPlateau several time
s to alternate high and low lr. The code is implemented with Pyt
orch. The total time of training for one fold is 1-2 hours, so t
he training of entire model is 6-9 hours. Almost all our models
were trained at kaggle, and the kernel is at this link.

Submission:
Use of 64 mels has allowed us also to shorten the inference time
. In particular, each 5-fold model takes only 1-1.5 min for gene
ration of a prediction with 8TTA. Therefore, we were able to use
 an ensemble of 15 models to generate the final submissions.
The final remark about the possible reason of the gap between CV
 and public LB: this paper (Task 1B) reports 0.15 lwlrap drop fo
r test performed on data recorded with different device vs the s
ame device as used for training data. So, the public LB data may
 be selected for a different device, but we never know if it is

```
true or not.
===
Thanks for sharing the method!!
Always learned a lot from you , @iafoss .
Nice new paper implement ! (Spectral augmentation)
Hope you get a good result for this competition !!
Options
You are welcome and thanks, u2 best luck at the stage 2

May I ask, have your team used any kind of minority class oversa
mpling? Have you applied MixUp with some probability or just aug
mented the whole dataset? Thanks for sharing and good luck in fi
nal shakeup!
Thanks, and good luck at the stage 2.
We have applied it to all samples in a batch, and we did not use
 any special consideration for any classes. The implementation i
s borrowed from here.
Thanks for sharing!
I should have tried 64 mels, it looks promising in your writing
;)

@ebouteillon, I wish u also the best luck. The idea suggested by
 @theoviel about 64 mels is really unexpected to work. However,
when I tried to switch back to 128 mels in our best setup later,
 I didn't see improvement. The thing could be also that for 128
mels ~100 epochs (in total) was not enough since stage1 Dnet mod
els we trained for 300-400 epochs.
Thanks so much @iafoss and all your team for the very detailed s
olution. I did try 4s (but with 128 mel) and had no improvement,
 perhaps I have to go back and redo my experiment :)

Very nice writeup @iafoss
I miss your awesome kernels.. There were great kernels in this c
ompetition especially those of @daisukelab and @mhiro2
I had almost the same experiments like yours (only part of what
you did).. I wished I had more time than the last 2 days for thi
s comp and those 2 days was hopeless with only 4 submissions , w
hich was surprising that with a good dose of luck it got into th
e silver range.
One question about your model arch. How did you think about chan
ging the arch into what you did? Is it try and see, or you had a
n idea that the model should look like this and not that?
I wish you and your team all the best in the 2nd stage. I have l
earned a lot from your kernels and from @suicaokhoailang and @ra
dek1 sharing in the past competitions.
@hwasiti, It is quite impressive that u could get that far just
within 2 days, hope u also get a good score at the stage 2. Rega
rding your question, the main reason why I focuses mostly on den
se connections (within the conv block and pyramid pooling) is th
at Dnet models worked much better than ResNet, ResNeXt, CBAM Res
NeXt, NasNet, etc. at stage 1. Also, concat pooling gave quite n
oticeable boost in my early testing. Later I tried other things
too, but they didn't work that well. The thing I didn't expect i
s that traditional 7x7 conv followed by pooling in the first lay
er doesn't really work here. Probably, if one just took Dnet121
and replace its first block, it could work quite well on 64 mels
```

```
, but I didn't have time to check it. And thank you for your bes
t wishes.
Thanks for sharing, I didn't even imagine decreasing mels down t
o 64.
And I agree with mixup is so efficient in audio task, "sounds ar
e transparent" - that's it.
I hope your team's good luck in 2nd stage.

Thanks, u2. And thank you for your kernels, I used one as a star
ting point for mel preprocessing.

@iafoss Thanks for sharing in detail. Maybe I should try to redu
ce mel to 64 and increase duration to 4s. Haven't thought about
reducing mel would gain better CV or LB

The key point with 64 mel is using small models with optimized a
rchitecture. Regular computer vision models require 128 mels, an
d also may be up-scaling as we did in our first attempts, that m
akes the models quite slow at training and inference. Meanwhile
with 64 mel setup we could reach 0.715 public LB score for a sin
gle 5-fold model trained within one kernel (~6 hours). Ensemblin
g boosted it to 0.739.

We will release the kernel used for training the models when pri
vate LB is available.

Great, really keen to see your kernel.

The kernel right now is available at https://www.kaggle.com/theo
viel/9th-place-modeling-kernel

@iafoss Great work. And congrats to be Kaggle Competition Master
. Every time I can learn a lot from your kernel. Also learned a
lot from your Airbus kernels.

Thanks so much, I'm happy to know that my kernels were useful.
```