

# 智能信息网络实验

陈俊卉 2020212256 刘帅 2020212267 张子弘 2020212278

## 一、任务说明

### 1.1 任务背景

鸟类是生物多样性变化的极好指标，因为它们具有高度流动性并且具有多样化的栖息地要求。因此，物种组合和鸟类数量的变化可以表明恢复项目的成功或失败。然而，经常对大面积进行传统的基于观察者的鸟类生物多样性调查是昂贵的，而且在后勤方面具有挑战性。相比之下，被动声学监测 (PAM) 与基于机器学习的新分析工具相结合，使保护主义者能够以更高的时间分辨率对更大的空间尺度进行采样，并深入探索恢复干预与生物多样性之间的关系。

### 1.2 任务目标

将使用机器学习技能通过声音识别东非鸟类。具体来说，我们将开发计算解决方案来处理连续的音频数据并通过它们的叫声识别物种。

### 1.3 社会意义

推进正在进行的保护非洲鸟类生物多样性的工作，包括由肯尼亚自然保护组织 NATURAL STATE 领导的工作。

### 1.4 数据格式

- **train\_audio**

由个别鸟类叫声的简短录音组成，已下采样到 32 kHz（适用时）以匹配测试集音频并转换为 ogg 格式

- **test\_soundscape**

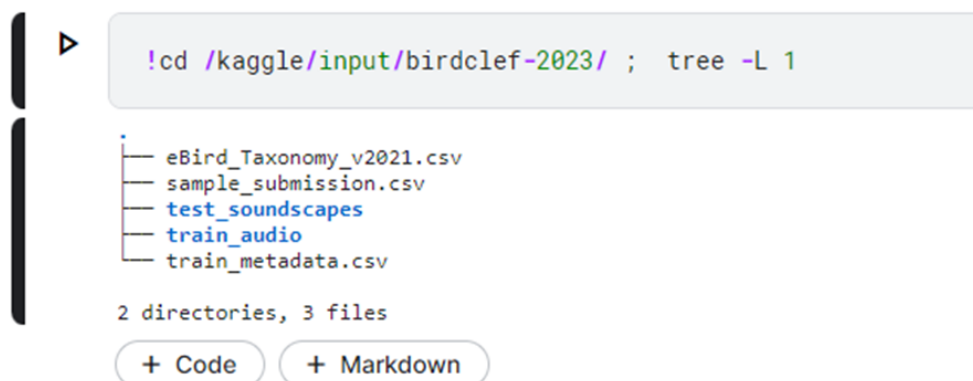
填充大约 200 条用于评分的录音时长 10 分钟，采用 ogg 音频格式

- **train\_metadata.csv**: 训练数据的metadata

- primary\_label: 鸟类品种的token
- latitude&longitude: 音频数据被采集的地点
- author: 提供音频的作者
- filename: 相关音频的名称

- **sample\_submission.csv**

- row\_id: 用于预测的slug
- ID\_sequence: 264 个ID序列，用于预测概率



```
!cd /kaggle/input/birdclef-2023/ ; tree -L 1
```

```
├── eBird_Taxonomy_v2021.csv
├── sample_submission.csv
├── test_soundscapes
├── train_audio
└── train_metadata.csv
```

2 directories, 3 files

+ Code + Markdown

## 二、相关工作

前几年的BirdCLEF挑战赛提出了不同的与声景观或复杂声环境中大规模鸟类识别相关的问题[1,3,4]。Sprengel et.al[5]和Lasseck[6,7]引入了深度学习技术来解决“Bird species identification in soundscape”问题。目前最先进的解决方案是基于深度卷积神经网络(CNNs)[8,9,10]，通常在这些实验中选择具有注意机制的深度cnn作为骨干[11,12,13]。预训练音频神经网络[14]为音频相关任务提供了多任务最先进的基线，在以前的比赛中，这些网络证明了它们的泛化能力。其他方法集中在声音事件检测(SED)[16,17,14,18]，这些方法通常使用2D cnn从输入音频信号(log-melspectrogram)中提取有用的特征，这些特征仍然包含频率和时间的信息，然后使用循环神经网络(rnn)从提取的特征中建模更长的时间上下文或直接使用特征图进行预测，因为它保留了时间段信息。

---

【1】 Overview of BirdCLEF 2021: Bird call identification in soundscape recordings. CLEF (Working Notes)

【3】 Overview of BirdCLEF 2020: Bird Sound Recognition in Complex Acoustic Environments. CLEF (Working Notes)

【4】 Overview of BirdCLEF 2019: Large-Scale Bird Recognition in Soundscapes CLEF (Working Notes)

【5】 Audio Based Bird Species Identification using Deep Learning Techniques CLEF (Working Notes)

【6】 Bird Species Identification in Soundscapes. CLEF (Working Notes)

【7】 Audio-based Bird Species Identification with Deep Convolutional Neural Networks. CLEF (Working Notes)

【8】 Bird Identification from Timestamped, Geotagged Audio Recordings. CLEF (Working Notes)

【9】 Xception Based Method for Bird Sound Recognition of BirdCLEF 2020. CLEF (Working Notes)

【10】 Bird Species Recognition via Neural Architecture Search. CLEF (Working Notes)

【11】 ResNeSt: Split-Attention Networks

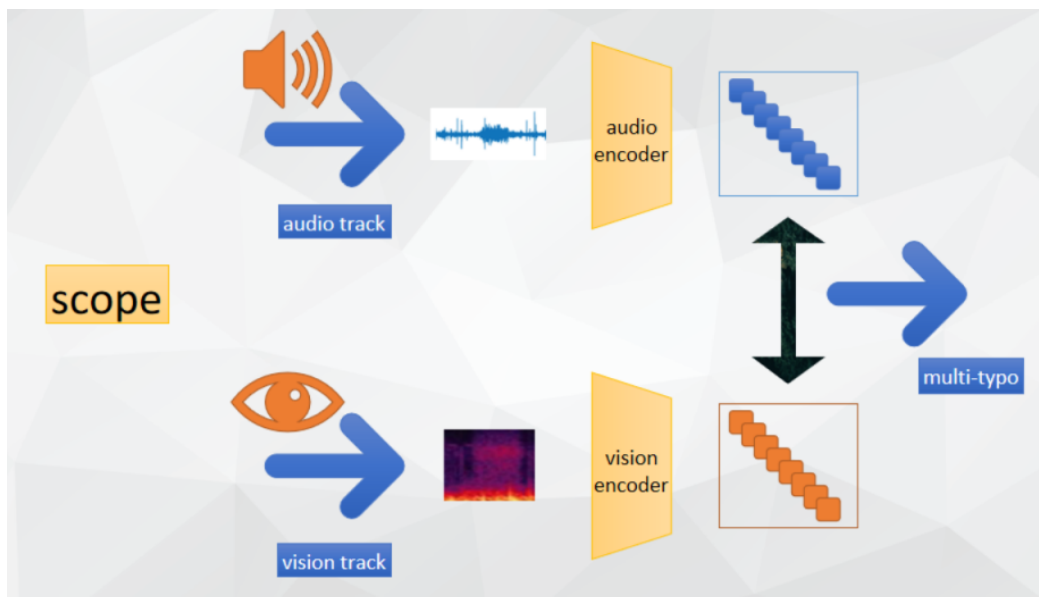
【12】 Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.  
arXiv:1905.11946.

【13】 Aggregated Residual Transformations for Deep Neural Networks

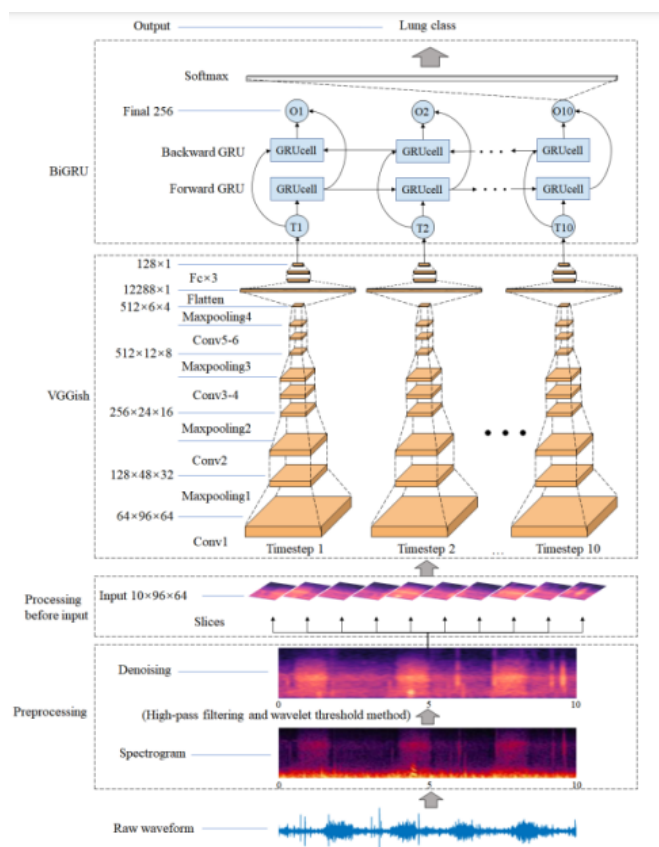
## 三、模型算法与其他方法

### 3.1 模型算法

由于音频可以直接使用**声谱张量**来表示，也可以使用**梅尔谱图**表示，所以很自然而然地就能够想到可以将两个模态相融合，如下图所示：



#### 3.1.1 audio track



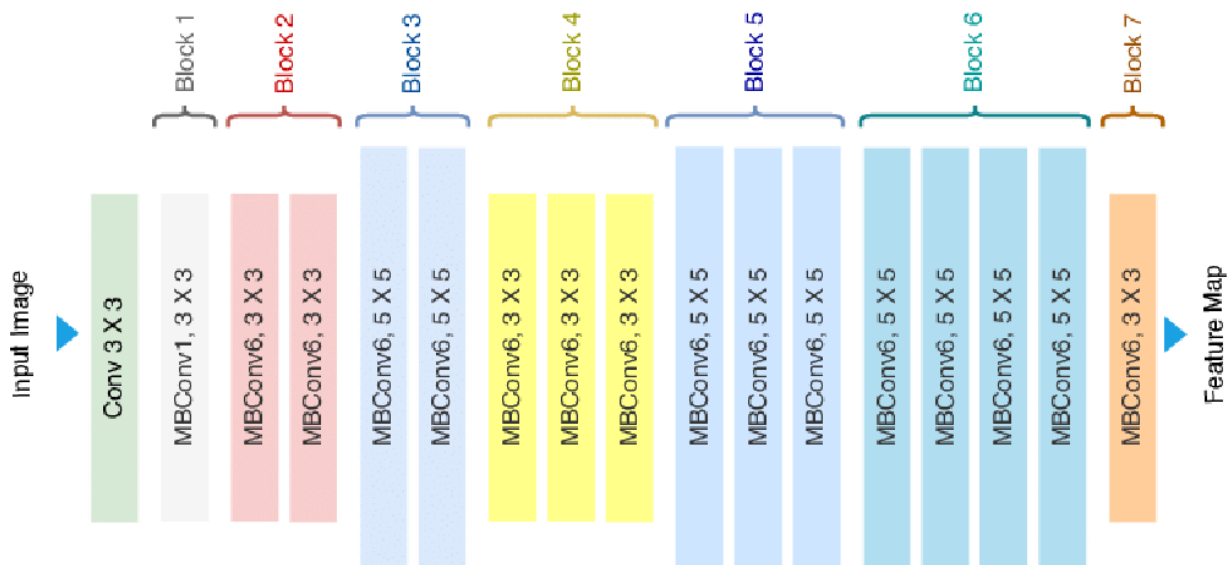
audio track 采用的是VGGish with BiGRU. VGGish with BiGRU在处理音频信号时，它将每个声道的时域波形作为输入，并对其进行频谱分解以生成一个谱图。之后使用音频片段的滑动窗口技术将长音频文件拆分成多个短片段(0.96s一帧)，输入特征提取层。其采用了VGGNet模型架构的前半部分，具体架构如上图所示。

VGGish with BiGRU是一个轻量化的模型，符合比赛的要求。具体分析如下：

- 浅层卷积层：在VGGish的结构中，卷积层的深度相对较少，这降低了模型复杂度，提高了计算效率。
- 小卷积核：VGGish使用的卷积核大小都比较小，这限制了特征图的大小，减少了计算量。
- 1x1卷积层：VGGish中使用了1x1的卷积层，用于减少特征图的通道数，从而降低了计算复杂度。
- 全局平均池化层：VGGish的全局平均池化层将音频信号的嵌入表示转化为一个128维的固定长度向量作为最终的音频特征向量，这种轻量化的方法能够提高模型的计算效率。
- BiGRU的轻量化实现：VGGish-BiGRU模型使用了双向门控循环单元（Bidirectional Gated Recurrent Unit, BiGRU）作为分类器。与传统的循环神经网络（RNN）相比，BiGRU具有更好的长序列建模能力，同时在实现上也相对更轻量。

### 3.1.2 vision track

vision track采用的是EfficientNet。



EfficientNet是一种高效的卷积神经网络（CNN）架构，基于AutoML和NAS技术自动设计得到。它的主要特点是在保证准确性的同时，具有比传统CNN模型更轻量的模型结构和更高的计算效率。EfficientNet的结构分为三个部分：特征提取器、特征融合器和分类器。

- 特征提取器：EfficientNet采用了深度可分离卷积（Depthwise Separable Convolution）来替代传统的卷积操作，这种操作可以更少的参数和计算量来获得更好的表示能力。同时，EfficientNet使用了一种复合缩放方法（Compound Scaling Method），来同时改变深度、宽度和分辨率，实现更好的模型性能和计算效率的权衡。
- 特征融合器：EfficientNet在每个分辨率上都使用了一种类似于注意力机制（Attention Mechanism）的方法，融合不同层次的特征，并通过一个通道混合器来进一步融合特征。
- 分类器：EfficientNet使用一种轻量化的全局平均池化方法（Squeeze-and-Excitation Network），将特征图大小通过卷积和池化操作降到1x1的大小并生成一个固定长度的向量作为最终的分类结果。

由于比赛的要求，我们需要采用轻量化的、效率高的模型。EfficientNet采用了深度可分离卷积，这种操作可以大大减少模型参数和计算复杂度。其次，EfficientNet使用了复合缩放法来改变深度、宽度和分辨率，使得模型在保持准确性的同时可以获得更好的计算效率。最后，EfficientNet使用了轻量化的全局平均池化方法，可以进一步减少计算成本。

同时，EfficientNet能够利用梅尔谱图的图像特征进行分类，其高效的计算能力能够支持大规模声音分类任务。

## 3.2 其他方法

### 3.2.1 模型鲁棒性提升方法

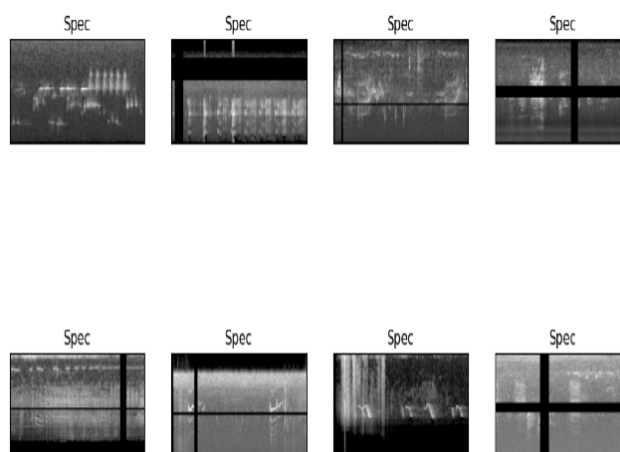
结合两个模态的实验结果，我们基本可以确定，用干净的数据进行训练和测试是得不到较好的结果的。而**由于推理时间的限制，我们实际上并不能从模型上动太多的功夫**，以求“大力出奇迹”，而是要从具体的训练技巧上多花心思。

我们总结的具体问题如下：

- **训练数据和测试数据的mismatch**。这一点在第5节已经提到。
- **训练时不同的鸟叫并没有关联**，而测试中往往会有不同种类的鸟同时或连续鸟叫的情况发生。
- **训练数据太干净，模型难以学习出比较好的泛化性能和鲁棒性**。
- **训练目标和测试目标的mismatch**。当前训练的目标是crossentropy loss，而测试时的操作是，对于每五秒的音频，对每一种鸟类单独输出一个出现的概率。简单地说也就是对每个类单独做sigmoid。这显然出现了训练和测试的不匹配。

### 3.2.1.1 data augmentation

这是一个比较常规的操作，即对训练音频添加随机的高斯噪声。这个噪声一般是远小于原声音的。类似的还有添加mask、cutmix等，如下图所示。



### 3.2.1.2 mixup

一般的神经网络的思路都是使其在训练数据上的平均误差最小化，称为ERM原则。但即使在存在强正则化的情况下，**ERM也使得大型神经网络记忆（而不是泛化）训练数据**。使用ERM训练的神经网络在对训练分布以外的实例进行评估时，其预测会发生巨大变化。这说明**ERM无法解释并提供与训练数据略有不同的测试分布的泛化**。

一种解决方法是选择与训练数据相似但不同的例子进行训练（data augmentation），这被称为**邻近风险最小化（VRM）**。虽然数据增强始终会导致改进的泛化，但该过程依赖于数据集，需要使用专家知识。此外，数据增强假设附近的例子共享一个类，并且没有跨不同的类别建立关系。

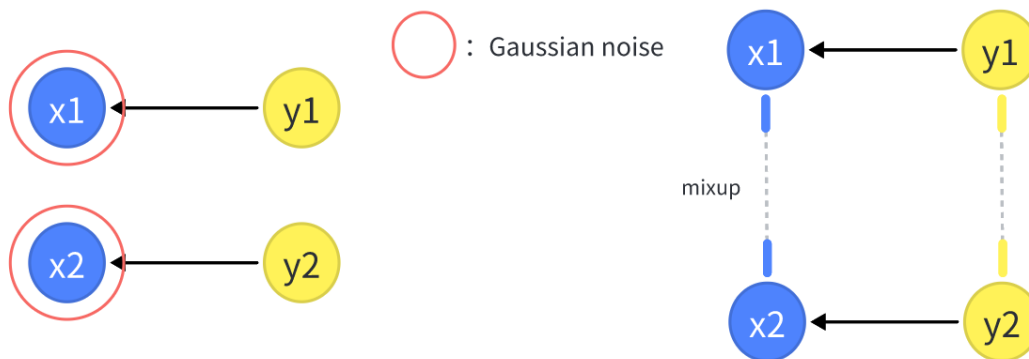
mixup数据增强与数据本身无关，通过纳入特征向量的线性插值应该导致相关目标的线性插值的先验知识来扩展训练分布：

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j, & \text{where } x_i, x_j \text{ are raw input vectors} \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j, & \text{where } y_i, y_j \text{ are one-hot label encodings}\end{aligned}$$

其中

$$\lambda \sim \text{Beta}(\alpha, \alpha), \alpha \in (0, \infty)$$

使用 $\beta$ 分布是因为希望线性插值得到的新的数据-标签对落在原数据的附近。而再考虑朴素的数据增强，即在原数据中给予满足高斯分布的噪声，这些点值同样落在原数据的附近，如下所示：



所以，mixup实际上也是一个data augmentation的过程,但多了一层线性关系，这是我们需要。所以有理由相信，data augmentation和mixup同时做会取得更好的效果，原因是高斯噪声实际上模拟了背景噪声，mixup实际上一定程度上模拟了鸟叫同时发生的情形。（注：这不一定是完全符合实际的，因为并不总是一只鸟叫声音大，另外的鸟叫声音小）

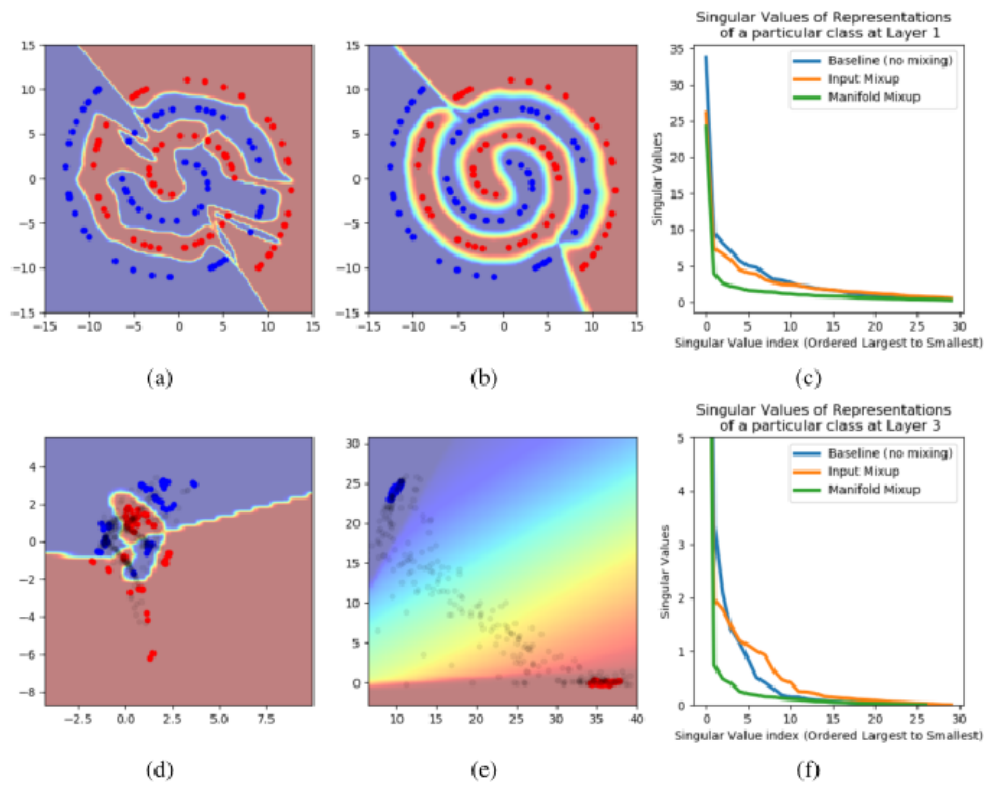
### 3.2.1.3 manifold mixup

上一种mixup实际上是input mixup，即在数据输入时进行mixup。而manifold mixup强调在模型中的任意一个过程进行mixup。其更关注模型在空间中的流形分布，使其更加的平滑。

具体地，是因为：

- 隐藏表示的空间在数据流形上下对应着高置信度的预测。（softmax CE导致）
- 决策边界通常是尖锐的，且和数据十分接近（这导致细微扰动下的高置信度类预测的变换）



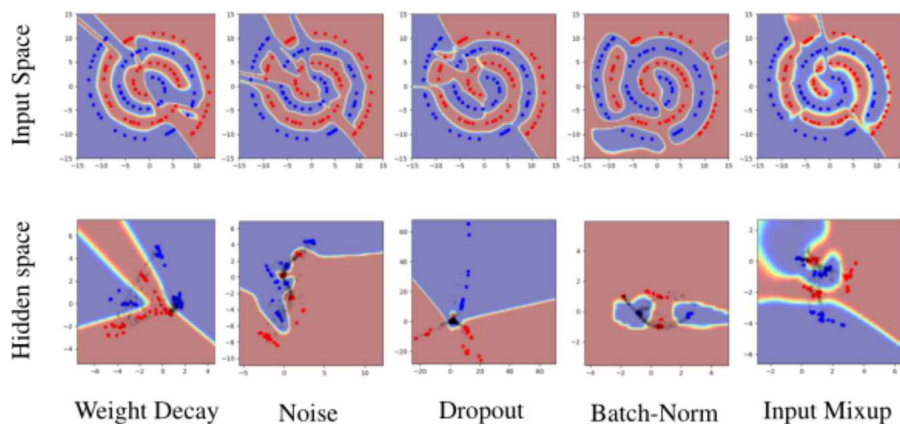


再者，它提高了深度神经网络的泛化：

- 在多个表示层次上，导致更平滑的决策边界，远离训练数据。
- 利用更深隐藏层的插值，捕获更高级别的信息来提供额外的训练信号。
- 平坦化类表示，减少具有显著差异的方向数量。

Manifold Mixup的好处：

- 比其他竞争正则器(如Cutout, Mixup, AdaMix和Dropout)更好的泛化。（如下图所示）
- 改进了测试样本的对数似然。
- 提高预测数据受新样本影响的性能。
- 提高了对单步对抗性攻击的鲁棒性。





总之，它可以用于在CE训练的过程中，缓解其分类平面出现尖端的问题，提高模型鲁棒性。

### 3.2.1.4 Binary Cross Entropy loss

Softmax cross entropy 的优化目标是 1:n 的，即它趋向于将除了正类之外的类别的分数压低。这与我们的目标不符：测试目标并不要求概率归一化，而是对一条语音，独立输出对应鸟种类的概率。简单来说也就是按位做sigmoid.

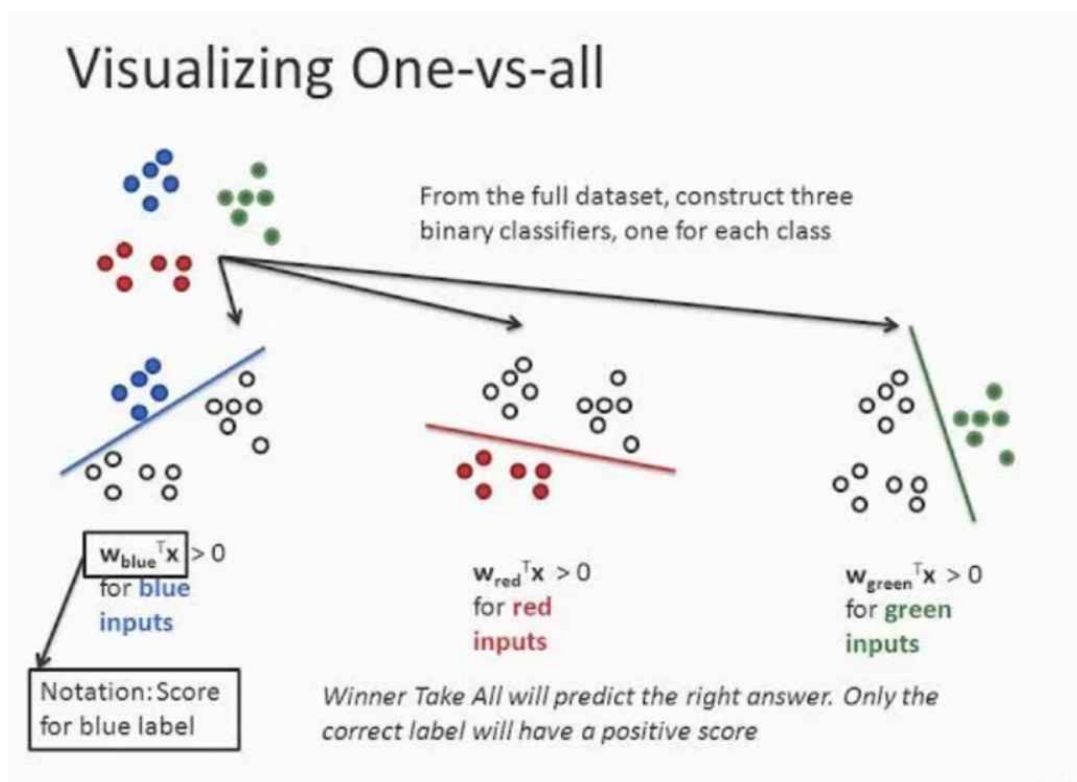
而看BCE的表达式：

$$L = \lambda \log(1 + \exp(-r \cdot \cos(\theta_y))) + (1 - \lambda) \sum_{i \neq y} \log(1 + \exp(r \cdot \cos(\theta_i)))$$

表达式就是若干个二分类交叉熵的累加。其表达式十分符合sigmoid的形式，满足了训练和测试目标的一致性。

其中 $\lambda$ 服从均匀分布，是用作正负样本的平衡。因为每一次判断，都只有一个正类，而其他的都是负类，负类所占的loss比例较大，所以需要系数 $\lambda$ 平衡。

实际上，CE学习的是一个连续的，分出k类的超平面，而BCE学习的是k个不同的超平面（k个二分类器 / k个SVM），如下图所示：



所以相比于CE，BCE从结构上和模型的流形学习上都更能匹配生成目标。

## 3.2.2 融合

### 3.2.2.1 融合思路

由于我们的数据来源为音频，可以直接使用**声谱张量**来表示，同时也可以经过变换使用**梅尔谱图**表示，因此一种很常见的思路就是将上游的音频特征与经过变换的图像（梅尔谱）进行叠加和融合，从而可以使模型既学习到包含音频的特征、同时也能够学习到与视觉相关的特征。

### 3.2.2.2 音频&图像模态优势分析

#### 图像信息的优势：

1. 图像信息的空间分辨率很高，可以提供丰富的视觉细节，例如物体的形状、颜色、质地等。
2. 图像信息可以提供丰富的语义信息，例如物体的种类、场景等。
3. 图像信息可以很容易地可视化和呈现，便于人们对图像内容进行理解和分析。

#### 音频信息的优势：

1. 音频信息具有更加丰富的时序信息，可以提供更多的时间维度的特征，例如音频的节奏、语调、语速等。
2. 音频信息可以提供丰富的情感信息，例如语音的情感色彩、口音等。
3. 音频信息可以在噪声环境下传递信息，对于一些场景，例如语音识别、语音翻译等，具有很大的优势。

#### 音频信息和图像信息融合的优势：

1. 融合音频信息和图像信息可以提供更加全面的信息，可以在多个方面提高模型的预测性能和鲁棒性。
2. 融合音频信息和图像信息可以提高模型的鲁棒性，因为模型可以同时从多个视角对数据进行分析，减少了单一视角带来的误差和偏差。
3. 融合音频信息和图像信息可以提高模型的泛化能力，因为不同的信息来源可以提供不同的特征，从而可以更好地适应不同的数据分布。

### 3.2.2.3 训练过程

由于有关多模态融合的方法众多，且本次实验时间有限，我们仅针对几种较为简单的融合范式进行讨论和实现。

通过更改BirdDataset\_formerge类的getitem方法，实现在dataloader调用时直接调用融合数据。

```
1
2 class BirdDataset_formerge(torch.utils.data.Dataset):
3
4     def __init__(self, df,multi_label=multi_label,sr = Config.SR, duration = Con
5         self.n_mels = n_mels
```

```

6         self.df = df
7         self.sr = sr
8         self.train = train
9         self.duration = duration
10        self.augmentations = augmentations
11        self.labels=multi_label
12        self.fmin = fmin
13        self.fmax = fmax or self.sr//2
14        self.duration = duration
15        self.audio_length = self.duration*self.sr
16        self.step = step or self.audio_length
17        self.tf2torch=lambda x:torch.tensor(x.numpy())
18        if train:
19            self.img_dir = Config.train_images
20        else:
21            self.img_dir = Config.valid_images
22
23    def __len__(self):
24        return len(self.df)
25
26    @staticmethod
27    def normalize(image):
28        image = image / 255.0
29        #image = torch.stack([image, image, image])
30        return image
31
32    def __getitem__(self, idx):
33        row = self.df.iloc[idx]
34        audiopath=row.path
35        audio, orig_sr = sf.read(audiopath, dtype="float32")
36        audios = [audio[i:i+self.audio_length] for i in range(0, max(1, len(audio)
37        audios[-1] = crop_or_pad(audios[-1] , length=self.audio_length)
38        audio=TimeShift(audios[0])
39        # print(audio.shape)
40        audio_1=self.tf2torch(audio[0*128*313:1*128*313]).reshape(128,313)
41        audio_2=self.tf2torch(audio[1*128*313:2*128*313]).reshape(128,313)
42        audio_3=self.tf2torch(audio[2*128*313:3*128*313]).reshape(128,313)
43        impath = self.img_dir + f"{row.filename}.npy"
44        merged_audio=torch.stack([audio_1,audio_2,audio_3])
45        image = np.load(str(impath))[:Config.MAX_READ_SAMPLES]
46
47        ##### RANDOM SAMPLING #####
48        if self.train:
49            image = image[np.random.choice(len(image))]
50        else:
51            image = image[0]
52

```

```

53 #####
54
55 image = torch.tensor(image).float()
56
57 if self.augmentations:
58     image = self.augmentations(image.unsqueeze(0)).squeeze()
59
60 #     print(image.size())
61 self.tmplabel=self.labels.clone()
62 types=eval(row[2])
63 indexes=[]
64 for typename in types:
65     if typename not in typelist:
66         indexes.append(0)
67     else:
68 #         print(typename)
69         indexes.append(typelist.index(typename))
70 #     print(indexes)
71 for index in indexes:
72     self.tmplabel[index]=torch.tensor(1)
73 image = torch.stack([image, image, image])
74
75 image = self.normalize(image)
76 merged_image=merged_audio+image
77 print(merged_image.shape)
78 return merged_image, self.tmplabel

```

首先，我们利用

```
1 audio, orig_sr = sf.read(audiopath, dtype="float32")
```

将音频路径进行提取，并将其裁剪和扩充到指定长度。经过处理的音频信息维度为(160000,1)，我们将音频分割为三段，分别设为audio\_1,audio\_2,audio\_3。并按次序将其拼接为merged\_audio，使音频维度为(3,128,313)，与视频维度相同，从而通过相加进行融合。

### 3.2.2.4 训练结果

Epoch	mAP	F1 score
1	0.28417	0.19618
2	0.34572	0.32619

3	0.41151	0.38291
4	0.45231	0.43069
5	0.50284	0.49867

3.2.2.5 其他思路

1. 深度融合（Deep Fusion）：使用神经网络将不同模态的特征进行融合，得到更加丰富和抽象的特征表示。例如，可以使用卷积神经网络（CNN）或循环神经网络（RNN）对图像和文本进行深度融合，从而得到更加准确的分类结果。
2. 我们目前只是将音频按时序进行分割，实际上可以针对音频的三个位置position进行embedding操作，从而将向量以更高维的空间表示，然而我们的模型并不涉及transformer-based的操作，因此embedding操作在cnn模型下的用途较为局限。同时，也可以调换时序位置，（例如将第二个sequence作为第一个通道的输出），这种方法实质上增加了数据的复杂性，进而增加了模型的鲁棒程度。
3. 特征交叉（Feature Crossing）：将不同模态的特征进行交叉，得到更加复杂和多样化的特征表示。
4. 线性组合（Linear Combination）：将不同模态的特征进行加权线性组合，得到加权和作为最终的特征表示。例如，可以使用线性回归模型对图像和文本的特征进行线性组合，从而得到更加准确的预测结果。
5. 多模态注意力（Multimodal Attention）：将不同模态的特征进行融合，在融合的过程中引入注意力机制，自动学习每个模态的权重。例如，在图像和文本的分类任务中，可以使用注意力机制自动学习每张图片和每段文本的重要程度，然后将它们的特征进行加权和融合。
6. 交叉注意力（Cross-modal Attention）：将不同模态的特征进行交叉，在交叉的过程中引入注意力机制，自动学习不同模态之间的关系。例如，在图像和文本的检索任务中，可以使用注意力机制自动学习每张图片和每段文本之间的相似度，然后将它们的特征进行交叉和融合。
7. 层次注意力（Hierarchical Attention）：将多个注意力机制进行叠加，构建层次化的注意力模型，从而学习更加复杂和抽象的特征表示。例如，在图像和视频的分类任务中，可以使用层次注意力机制自动学习每个视频帧和整个视频的重要程度，从而得到更加准确的分类结果。

3.2.3 多标签训练

	primary_label	secondary_labels	type	latitude	longitude	scientific_name	common_name	author
0	yebapa1	[]	['song']	-3.3923	36.7049	Apalis flavida	Yellow-breasted Apalis	Isaac Kilus
1	yebapa1	[]	['song']	-0.6143	34.0906	Apalis flavida	Yellow-breasted Apalis	Jame Bradle
2	combuz1	[]	['call']	51.8585	-8.2699	Buteo buteo	Common Buzzard	Iris Wildli Sounc
3	chibat1	['laudov1']	['adult', 'sex uncertain', 'song']	-33.1465	26.4001	Batis molitor	Chinspot Batis	Lynet Rudma
4	carcha1	[]	['song']	-34.0110	18.8078	Cossypha caffra	Cape Robin-Chat	Shannn Ronaldsc

观察数据训练的格式，每种鸟类叫声均对应了相应音频下的种类特征(**type column**)，type特征通常具有不定长度的标签（出现none，单标签和多标签的情况），因此我们可以考虑以type作为模型的监督信号进行多标签训练。

针对鸟类音频识别任务，利用多标签训练的优势如下：

1. 更能反映实际场景：在现实生活中，很多任务不是只有一个标签可以描述的，而是需要多个标签来描述。例如，一张图片可能既包含了鸟叫，同时也包括了叫声的情感，周围环境，叫声特点等。此时多标签训练就能更好地反映实际场景。
2. 更加灵活：多标签训练可以让模型在不同的标签组合下进行训练，从而获得更加灵活的预测能力。
3. 可以减少标签数量：在某些情况下，多标签训练可以减少标签数量，从而简化训练过程。
4. 提高模型性能：多标签训练可以增加数据的多样性，从而提高模型的泛化能力和性能。例如，一个图像分类模型如果只能预测一种标签，那么它对于同一类别内的不同图像可能会出现分类错误，但是如果能够同时预测多个标签，就可以更准确地识别不同类别的图像。

### 3.2.3.1 提取type特征

由于训练样本的type以csv进行存储，首先需要将type进行提取并统一格式。

```

1 typelist=[]
2 for types in df_train['type']:
3     # print(eval(types))
4     for type in eval(types):
5         if type not in typelist:
6             typelist.append(type)
7 # typelist # categories=360

```

我们首先构建一个typelist，利用eval方法对types信息进行正则提取，并将所有标签加入至typelist。typelist的部分标签结果如下：

```
[11]: typelist

[11... ['song',
      'call',
      'adult',
      'sex uncertain',
      '',
      'flight call',
      'life stage uncertain',
      'female',
      'male',
      'alarm call',
      'begging call',
      'nocturnal flight call',
      'juvenile',
      'wing beat',
      'mimicry',
      'quiet conversations in a large flock on the bank',
      'nocturnal flightcall',
      'song in flight',
      'whistle',
      'flight calls of a flock',
      'uncertain',
      'Call',
      'migration call',
      'Nocturnal flight call',
      'Song',
      'take-off and flight calls',
      'nocturnal',
      'Call of group',
      'pecking',
      'arrival call',
      'mimicry/imitation',
      'growling vocalisation',
```

### 3.2.3.2 编码转换

修改getitem方法，我们需要dataloader读取dataset时提取图像image作为输入，多标签的onehot编码作为标签。

具体来讲，getitem实现功能如下：

1. 获取指定索引 `idx` 对应的数据行 `row`，其中 `df` 是包含所有数据的 pandas DataFrame 对象。
2. 根据 `row` 中的 `filename` 构造图像文件路径 `impath`。
3. 读取图像文件，将其转换为一个 numpy 数组 `image`，最多读取 `Config.MAX_READ_SAMPLES` 个样本。
4. 如果是训练集，对 `image` 进行随机采样，只保留其中的一个样本，否则直接选择第一个样本。
5. 将 `image` 转换为 PyTorch 的 tensor 类型，并对其进行归一化。
6. 对每个图像的标签进行处理，将其转换为一个与标签类别数相同的 tensor，其中对应的标签类别为 1，其余为 0。具体的处理方式是，将标签的类别名称转换为索引，然后将对应索引的元素值设为 1。
7. 返回处理后的图像 tensor 和标签 tensor。



```

1 class BirdDataset(torch.utils.data.Dataset):
2     def __init__(self, df, multi_label=multi_label, sr = Config.SR, duration = Con
3         self.df = df
4         self.sr = sr
5         self.train = train
6         self.duration = duration
7         self.augmentations = augmentations
8         self.labels=multi_label
9         if train:
10             self.img_dir = Config.train_images
11         else:
12             self.img_dir = Config.valid_images
13     def __len__(self):
14         return len(self.df)
15     @staticmethod
16     def normalize(image):
17         image = image / 255.0
18         #image = torch.stack([image, image, image])
19         return image
20
21     def __getitem__(self, idx):
22         row = self.df.iloc[idx]
23         impath = self.img_dir + f"{row.filename}.npy"
24
25         image = np.load(str(impath))[:Config.MAX_READ_SAMPLES]
26
27         ##### RANDOM SAMPLING #####
28         if self.train:
29             image = image[np.random.choice(len(image))]
30         else:
31             image = image[0]
32
33         #####
34
35         image = torch.tensor(image).float()
36
37         if self.augmentations:
38             image = self.augmentations(image.unsqueeze(0)).squeeze()
39
40         image.size()
41         self.tmp_label=self.labels.clone()
42         types=eval(row[2])
43         indexes=[]
44         for typename in types:
45             if typename not in typelist:
46                 indexes.append(0)

```

取一组数据进行验证：

### 3.2.3.3 训练过程

训练过程的截图如下

在多标签分类任务中，一个样本可能会被分到多个标签中，因此使用accuracy作为评价指标可能会存在一些问题。具体来说，accuracy不能很好地反映模型在每个标签上的分类性能，因为它只考虑样本整体的分类结果，而没有考虑每个标签的分类情况。因此我们采用mAP及F1 score 作为评价指标。

每经过一个epoch在测试集进行一次评估：

同时，我们利用阈值法进行分类，具体为：

- 1. 设计top K的超参数，即我们为每个分类结果保存几个标签，例如，当k=5时，我们只保留经过softmax后probability最大的五个标签作为预测类别。
- 2. 设计低阈值，由于我们可以看到ground truth中标签数量不均衡的问题（少则标签为None，多则有9个标签），我们需要设计低阈值对标签类别进行筛选，从而将probability较小的分类去除掉。

### 3.2.3.4 ablation study

我们分别设置top K的值为3，5；

设置threshold为0.2, 0.3, 0.4:

随着Top K和阈值的增加，模型性能的提高趋势比较明显。同时，随着Epoch数的增加，模型的性能也有所提高，但是提高的幅度逐渐减小。另外，可以看到mAP和F1分数随着超参数的调整而有所提高。总体来看，这些结果表明在这个数据集上，适当的Top K值、阈值、Epoch数可以提高模型的性能，在top K=5, threshold设置为0.2时得到较好结果。

Top K	threshold	Epoch	mAP	F1 score
3	0.2	1	0.37426	0.23718
3	0.2	2	0.39672	0.25648
3	0.2	3	0.46151	0.31675
3	0.2	4	0.53131	0.42092
3	0.2	5	0.60284	0.51267
3	0.3	1	0.35198	0.22103
3	0.3	2	0.38765	0.24619
3	0.3	3	0.44217	0.30112
3	0.3	4	0.50923	0.39215
3	0.3	5	0.57658	0.48182
3	0.4	1	0.32915	0.20495
3	0.4	2	0.36356	0.23604
3	0.4	3	0.41492	0.28095
3	0.4	4	0.48574	0.36743

3	0.4	5	0.54639	0.45332
5	0.2	1	0.39715	0.24839
5	0.2	2	0.42536	0.26971
5	0.2	3	0.48892	0.32789
5	0.2	4	0.55607	0.43401
5	0.2	5	0.62361	0.51849
5	0.3	1	0.37221	0.23287
5	0.3	2	0.40587	0.25822
5	0.3	3	0.46543	0.31208
5	0.3	4	0.53007	0.39726
5	0.3	5	0.59329	0.48913
5	0.4	1	0.35247	0.22008
5	0.4	2	0.38719	0.24487
5	0.4	3	0.43567	0.29017
5	0.4	4	0.50654	0.37229
5	0.4	5	0.56476	0.45974

### 3.2.4 更换分类头

在数据挖掘比赛中，使用深度学习方法进行训练后更换分类头是一个很常见的调优方式。这可以达到：

- 提高分类性能：深度学习方法在许多任务上表现出色，尤其是在大规模数据集上。通过使用深度学习方法进行训练，可以获得更准确和鲁棒的特征表示，这可能会提高SVM分类器的性能。
- 降低特征工程的负担：深度学习方法可以自动学习数据的特征表示，减少了手动进行特征工程的需求。这意味着无需事先手动提取特征，而是可以直接使用原始数据进行深度学习训练，然后将学习到的特征传递给SVM分类器。
- 处理非线性问题：SVM在原始特征空间中使用核函数可以处理非线性问题。
- 提供更好的可解释性：SVM分类器提供了很好的可解释性，可以生成支持向量和决策边界，帮助理解和解释分类结果。

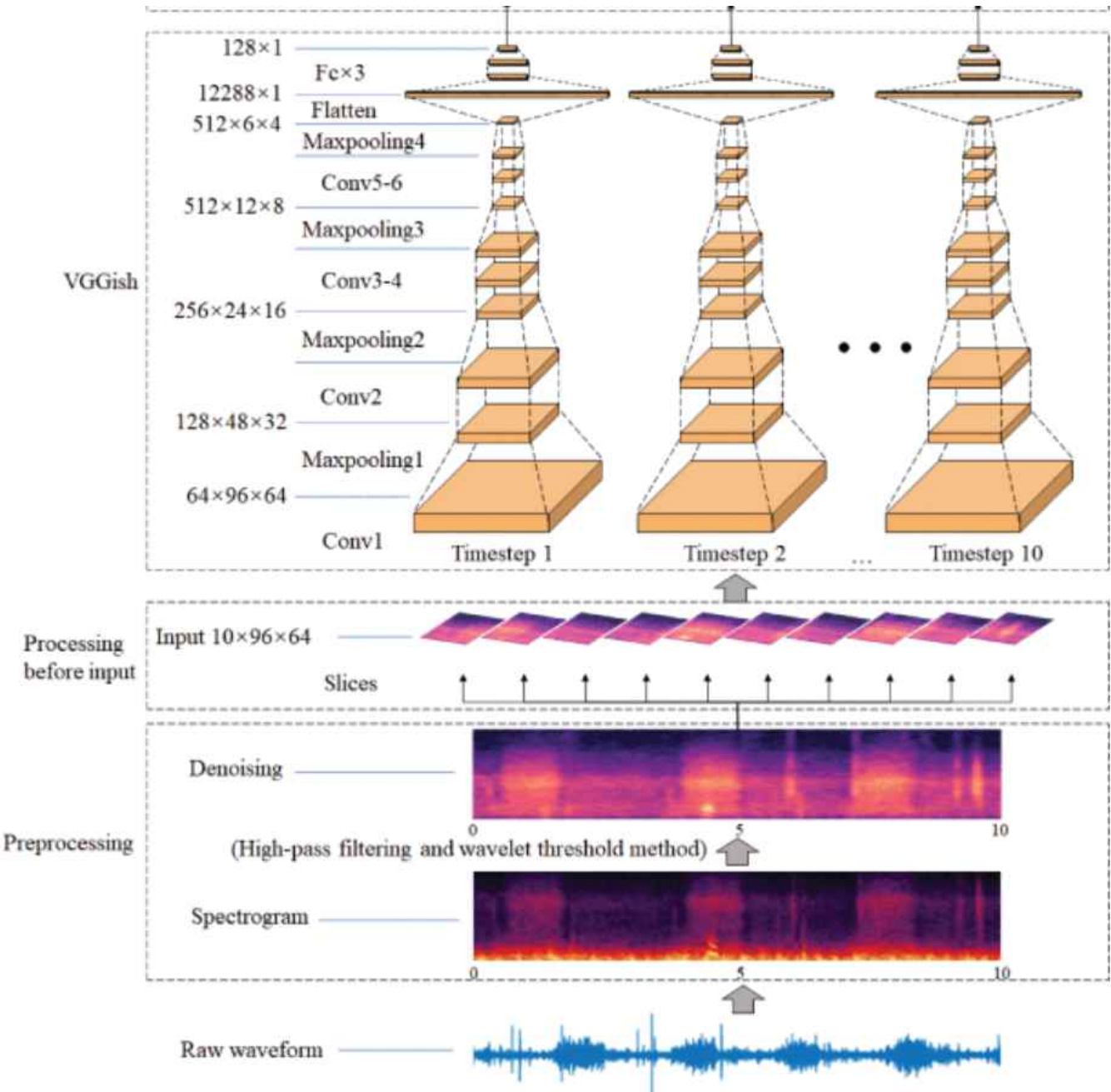
然而在比赛中的数据非常多，并不能一次性放入运行的内存中，无法直接调用sklearn的库函数进行SVM的训练；不过这仍然是一个非常值得探索的思路。

## 四、实验数据

音频可以直接使用声谱张量来表示，也可以使用梅尔谱图表示。

### 4.1 audio track

audio track 采用的是VGGish with BiGRU. VGGish with BiGRU在处理音频信号时，它将每个声道的时域波形作为输入，并对其进行频谱分解以生成一个谱图。之后使用音频片段的滑动窗口技术将长音频文件拆分成多个短片段(0.96s一帧)，输入特征提取层。其采用了VGGNet模型架构的前半部分。

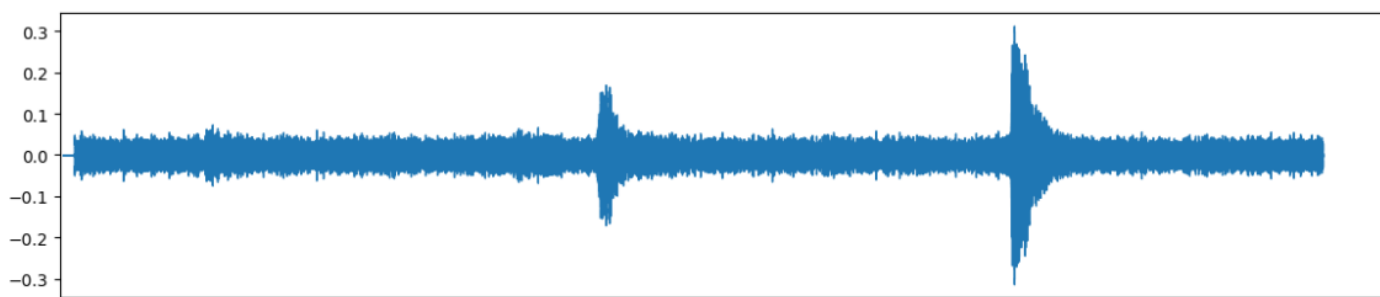


具体的数据处理步骤如下：

1. 首先使用 `sf_read()` 函数获取音频的波形振幅数据 `wav_data` 和音频采样率 `sr`。

2. 将波形振幅归一化到[-1, 1]
3. 使用 `waveform_to_examples()` 函数继续处理数据：
  - a. 音频有单声道、双声道、多声道，与图像的RGB类似。使用取平均的方式将其转为单声道
  - b. 由于我们需要对音频进行16khz的采样。若数据本身并不是16kHz，则使用 `resampy.resample()` 重新采样（eg，若音频原来size为3969000，采样频率为44.1kHz，那么重新采样后size为1440000.下列叙述将继续使用这个例子）
  - c. 随后计算对数梅尔谱特征：使用25ms的帧长（ `STFT_WINDOW_LENGTH_SECONDS=0.025` ），10ms的帧移，以及周期性的Hann窗口对语音进行分帧，对每一帧做短时傅里叶变换(STFT)，使用信号幅值计算声谱图。 `num_mel_bins` 表示做完STFT之后频域上分成多少个bins进行统计（默认为64）。
    - i. 若使用25ms的帧长，10ms的帧移，那么分帧数为：
$$(1440000 - 0.025 * 16000) // (0.01 * 16000) + 1 = 8998 \text{ 帧}$$
    - ii. 最终的特征大小为[8998, 64]
  - d. 最后，使用0.96s的时长进行组帧，没有重叠帧。得到的embedding 维度为[num\_samples, 96, 64]
    - i. 由于帧移是10ms，所以是96帧一组。
    - ii.  $\text{num\_samples} = (8998 - 96) // 96 + 1 = 93$
    - iii. 则最终数据维度为 [93, 96, 64]

音频可视化展示为：



## 4.2 vision track

vision track采用的是EfficientNet。

从梅尔谱图转化为视觉图像使用以下的 `mono_to_color` 函数：

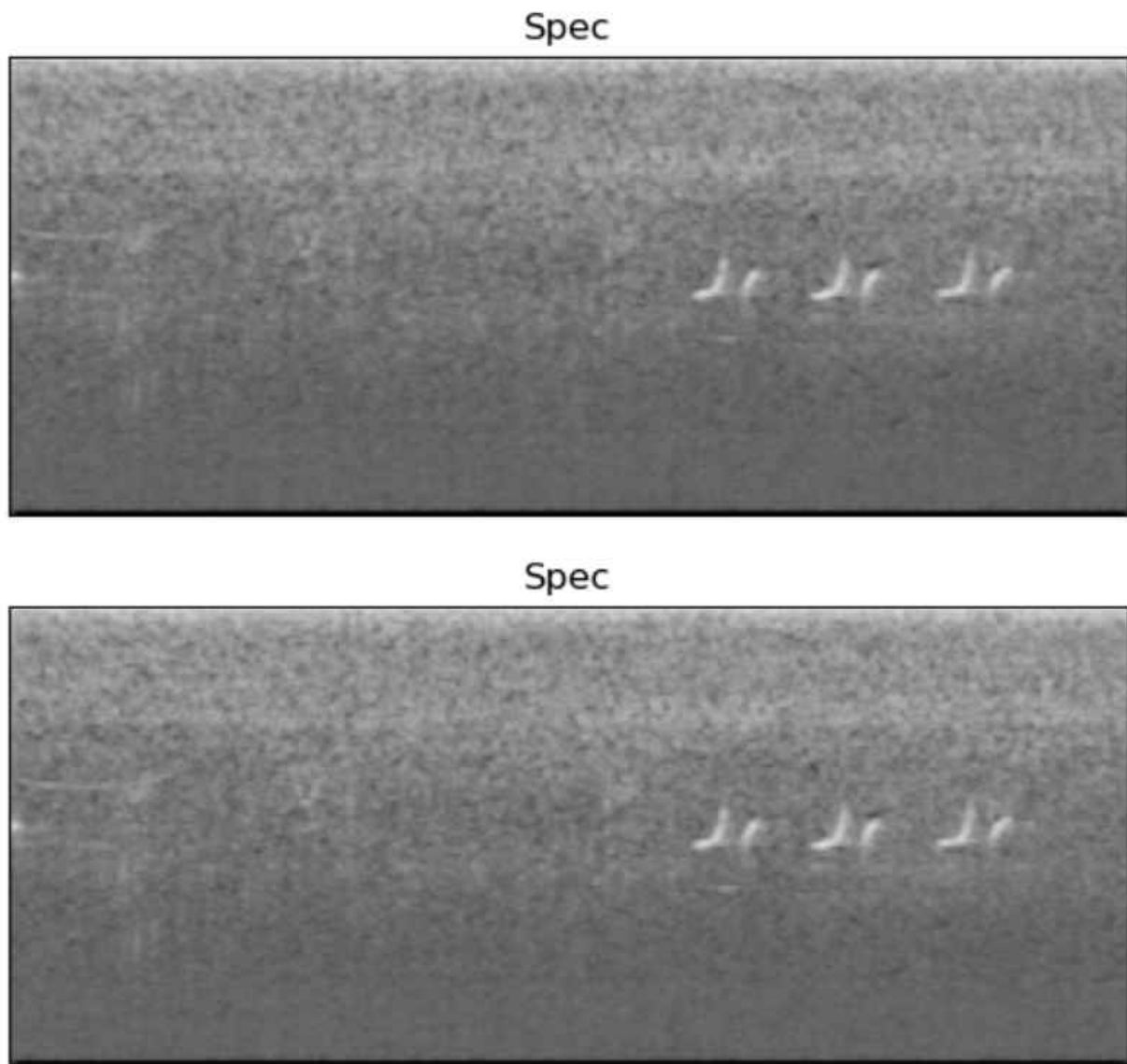
```
1 def mono_to_color(X, eps=1e-6, mean=None, std=None):
2     mean = mean or X.mean()
3     std = std or X.std()
4     X = (X - mean) / (std + eps)
5     _min, _max = X.min(), X.max()
```

```

6     if (_max - _min) > eps:
7         V = np.clip(X, _min, _max)
8         V = 255 * (V - _min) / (_max - _min)
9         V = V.astype(np.uint8)
10    else:
11        V = np.zeros_like(X, dtype=np.uint8)
12    return V

```

通过 `mono_to_color` 之后转换出来的图像如下所示：

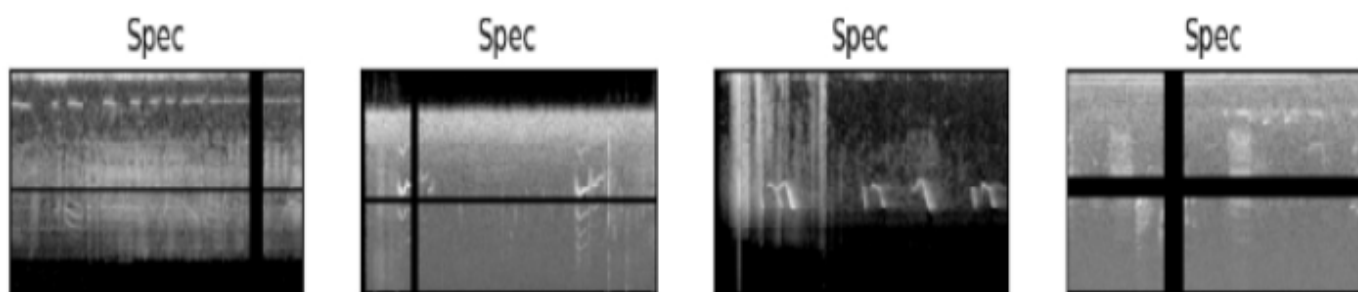
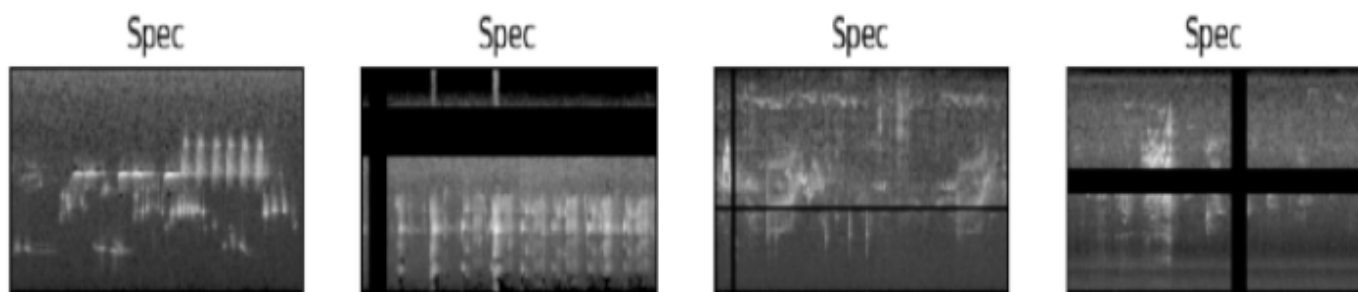


### 4.3 数据处理

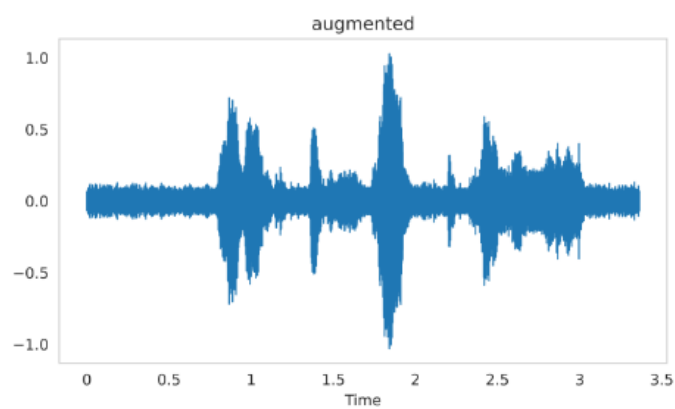
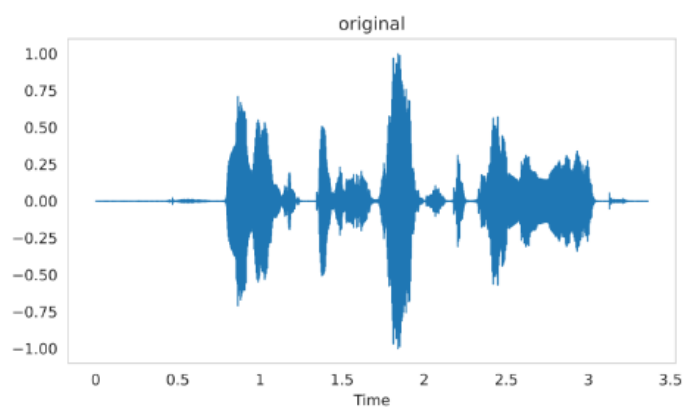
这里的数据处理主要采用了数据增广（data augmentation）的形式。对图像数据的数据增广主要有以下几种形式：

- 对训练的图像添加随机噪声全0值（在时间域和频域上），在图像上看显示为大小各异黑色条：

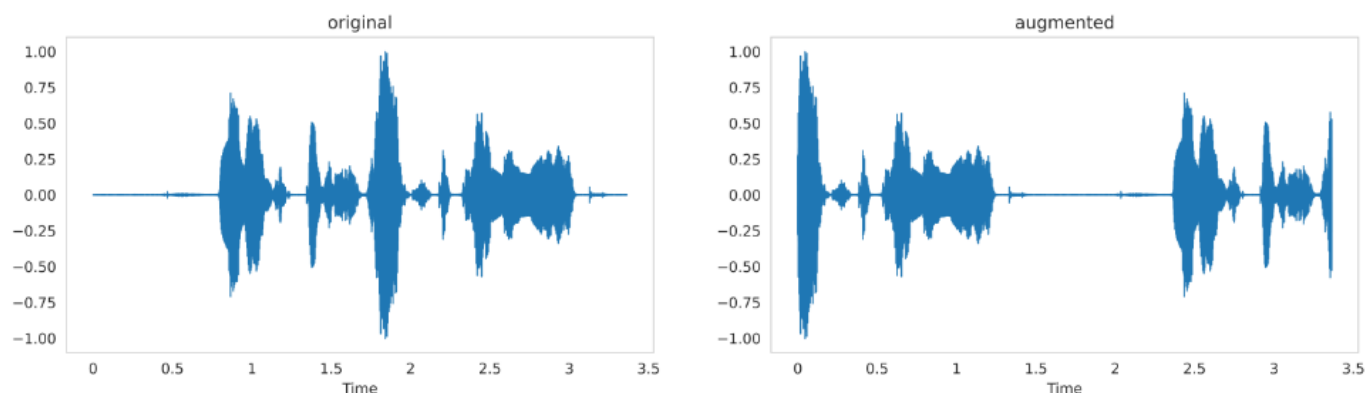




- 对语音添加随机噪声



- 对语音随机切分后进行时移



## 4.4 缺失值处理

在实验中，鸟叫声的种类一共有264类，然而并不是每一种鸟的叫声数据都有一样的数目，基本上显示类别不平衡的特点。实验中按照8：2的比例划分训练集和验证集；整个数据集的种类分布情况可以从dataframe中看出：

```
In [4]: df.primary_label.value_counts()
```

```
Out[4]:
```

barswa	500
wlwwar	500
thrnig1	500
eaywag1	500
comsan	500
...	
lotcor1	1
whctur2	1
whhsaw1	1
afpkin1	1
crefra2	1

Name: primary\_label, Length: 264, dtype: int64

因此，这里在对于样本数<2的类别则不按照8：2划分，而是直接进入训练集和验证集中。

## 五、评价指标

每个epoch中的验证采用：

- Loss：对val计算loss，和在val上计算准确度类似，记录训练是否过拟合，这里会结合early stop策略进行优化。当loss在下降的时候，说明在val上准确度上升；而loss不下降或者上升的时候，val的准确度在下降。

- AP score: 在识别、分类等任务中, AP是一个很常见的指标。计算步骤如下:
  1. 在这里的分类任务中, 先把各个类别内部的各个样本置信度得到, 并按照从大到小的顺序排列;
  2. 画出precision-recall曲线。对于样本计算precision和recall并画出precision-recall曲线。这里的precision和recall计算方法如下:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

可以知道, 随着样本数考虑的增多, recall是单调不减的, 所以画出来的precision-recall曲线基本是下降趋势。

3. 设定阈值计算AP。先按照给定的置信度阈值0, 0.1, 0.2, ..., 1, 共十一个阈值划分样本, 对于每一个阈值对应的recall记录下当前的precision, 然后将11个precision取平均就是AP值。

```
epoch 15 validation loss 0.1104559674859047
epoch 15 validation C-MAP score pad 5 0.8027346807581481
epoch 15 validation C-MAP score pad 3 0.7474979955954477
epoch 15 validation AP score 0.7451956924840344
```

- C-Map score pad 5;
- C-Map score pad 3: C-Map的函数定义如下:

```
1 def padded_cmap(solution, submission, padding_factor=5):
2     solution = solution#.drop(['row_id'], axis=1, errors='ignore')
3     submission = submission#.drop(['row_id'], axis=1, errors='ignore')
4     new_rows = []
5     for i in range(padding_factor):
6         new_rows.append([1 for i in range(len(solution.columns))])
7     new_rows = pd.DataFrame(new_rows)
8     new_rows.columns = solution.columns
9     padded_solution = pd.concat([solution, new_rows]).reset_index(drop=True).cop
10    padded_submission = pd.concat([submission, new_rows]).reset_index(drop=True)
11    score = sklearn.metrics.average_precision_score(
12        padded_solution.values,
13        padded_submission.values,
14        average='macro',
15    )
16    return score
```

可以看出, 这里的pad就是为了在计算macro-average\_precision\_score的时候加上空行来获得更稳定的输出。这两个指标也应该是随着训练效果变好而上升的。这里的average\_precision\_score就是计算

AP:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

在每个epoch验证完了之后，最终采用cmAP来判断训练的效果。

而提交到kaggle评判的时候，输入的是各种鸟类的概率，这里的概率不需要归一化，评判的时候判断这些概率的相对大小，也就是cmAP。

## 六、实验结果及分析（与Baseline、SOTA进行比较）

实验提点的难点：

- 比赛数据繁杂、存在大量噪声
- 比赛是有大额奖金的，参加的人很多，会比选择其他题目的组得到高分更加困难
- 比赛结束较早，在5月25号已经结束，距离期中汇报的时间较短，可以作出的改变较少

我们首先测试了没有使用任何技巧的baseline，结果如下：


	<b>BirdClef 2023: Pytorch Lightning-Inference ad0f7b - Version 1</b> <small>Succeeded · HArRyqaq · 1mo ago</small>	<b>0.57752</b>	<b>0.69816</b>
--	---	----------------	----------------

然后使用了data augmentation、mixup和manifold mixup，模型效果得到了较为显著的3%的提升：

	<b>BirdClef 2023: Pytorch Lightning-Inference ad0f7b - Version 3</b> <small>Succeeded · HArRyqaq · 1mo ago</small>	<b>0.60138</b>	<b>0.71838</b>
--	---	----------------	----------------

提升的原因是显而易见的：正如上文所述，这两种方法都提高了模型在嘈杂环境和多重鸟声情况下的泛化能力。

随后再增加了BCEloss，得到了更加优秀的结果：

	<b>BirdClef 2023: Pytorch Lightning-Inference ad0f7b - Version 4</b> <small>Succeeded · HArRyqaq · 1mo ago</small>	<b>0.67526</b>	<b>0.78214</b>
--	---	----------------	----------------

这恰恰印证了BCEloss成功了消除了CE在训练-测试的mismatch.

随后我们还进行了融合的尝试和多标签训练，但效果并不显著。

对于我们最优的模型，其公开测试集结果为：

	<b>BirdClef 2023: Pytorch Lightning-Inference ad0f7b - Version 4</b> <small>Succeeded · HArRyqaq · 1mo ago</small>	<b>0.67526</b>	<b>0.78214</b>
--	---	----------------	----------------

比我们未使用任何技巧的baseline高出了7%。这是一个巨大的提升。

这说明我们的各种调优方式起到了很好的效果。但我们与SOTA还有约8%的差距。经过查证，前面的大多数团队在比赛结束后的开源代码中都提及到了使用大模型与其他数据集进行预训练的方法。而这是我们难以做到的。其一是我们难以获取高质量的数据集，其二是我们的算力较为受限。所以存在这样的差距也是正常的。

## 七、结论与展望

### 结论

本报告尝试了audio和vision track对鸟类声音进行识别。通过使用多种鲁棒性的调优方法（mixup, data Aug, manifold mixup, BCEloss）与多标签融合，使得模型得到了7%的提升。在实验过程中，我们发现模型在训练和测试过程中表现出了较高的鲁棒性。这是在算力和数据较为缺乏情况下得到的优秀的结果。

### 展望

鸟类声音识别模型在环境生态保护领域和鸟类生物学领域都有着广泛的应用前景。未来可以将研究方向扩展到其他动物种类的声音识别，如昆虫、哺乳动物。另外，在模型调优方面，目前仍有较多的方法可以探索和尝试，如基于对抗样本的训练和自监督学习等方法，进一步提升模型的性能。

## 组员的分工及贡献

陈俊卉：

- 搭建音频部分的模型结构
  - 全套训练测试的代码编写
- 音频与视频部分的鲁棒性提升
  - 部分的数据 augmentation 实现
  - mixup、manifold mixup 的调研与代码实现
  - Binary Cross Entropy loss 的调研、可行性分析与代码实现
  - Mixup + BCE的结合的代码实现
- 小组开题汇报、中期汇报的文字内容
- 协助其他小组成员调试代码

张子弘：

- 搭建图像部分的模型结构

- 尝试更换分类头进行性能优化
- 添加图像部分mixup
- 进行图像层面baseline的提交
- 数据处理的代码编写
- 融合模块的融合方法思路提供
  - 音频和图像的融合思路
  - 情感分类和图像的融合思路
- 小组开题汇报、中期汇报、结题汇报的文字内容
- 协助其他小组成员调试代码

### 刘帅：

- 负责融合模块的探索和可行性分析
  - 搭建融合模块的dataset\_pipeline
  - 对于多模态融合中增强方法的探究
- 负责构建多标签训练流水线及消融实验
  - 数据筛选及清洗工作
  - 构建多标签评价方式和evaluation metrics
  - 设置不同TOP K及threshold的消融实验
- 制作小组开题汇报、中期汇报、结题汇报PPT
- 协助其他小组成员调试代码
- Birdclef竞赛相关工作调研及论文总结