Implementation of a Coarse-to-Fine Learning Pipeline on Audio Classification on SOTA Neural Network architectures

Pranav Gupta
RA2111003011091
B2
Neurofuzzy and Genetic Algorithms(18CSE352T)

Introduction

Environment sound classification has been a well-researched area in signal processing. Traditional methods have focused on fully supervised ways to solve the problem and have shown the greatest performance. However, there has been an increasing interest in recent times to utilize the potential of semi and self-supervised ways to improve the model performance with lesser data. While most semi-supervised methods focus on utilizing unlabelled data along with labeled data, self-supervised methods attempt to learn intermediate representation via pretext tasks or contrastive learning. However, both approaches require a large amount of unlabelled data to improve performance. In this work, a novel semi-supervised framework is proposed that utilizes label ontology-based hierarchy to learn semantic representation by defining a novel pretext task. In the pretext task, the model tries to predict coarse labels defined by a Large Language Model(LLM) based on label ontology. The trained model is further fine-tuned in a supervised way to predict the actual classes. Our proposed novel semi-supervised framework achieves an accuracy improvement in the range of 5% to 12% over baseline across three datasets namely UrbanSound8K, ESC10, and ESC50. You can find the code on GitHub.

Background

Due to its diverse and complex nature, Environmental Sound Classification (ESC) presents a substantial challenge in the fields of digital forensics, machine learning, and signal processing. Sound classification has different applications, including bird sound classification, audio-visual systems, smart city noise detection, healthcare monitoring, and others. The importance of sound classification is not limited to just human sounds, but it also includes other environmental sounds (ESC). This can be related to other domains like music instrument classification, speech recognition, and sound event detection, all of which show certain similarities with ESC. It still stands out with its unique challenges because of the unstructured data and low SNR that result from large distances between the sound sources and the recording devices.

Historically, techniques of feature extraction such as Wavelet features, Mel Frequency Cepstral Coefficients (MFCC), Gammatone features, and Chroma features have been applied for the application of ESC, even though these approaches have heavy computational costs. For some years now, deep learning methods like deep neural networks (DNN), convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory (LSTM) networks have shown potential in ESC by using spectrogram representations for better accuracy.

Although deep learning models manage to provide a higher level of performance, they require large annotated datasets and human guidance during training. In an attempt to deal with this problem, self-supervised learning (SSL) has evolved as a promising alternative, allowing the learning of representations that are useful in an unlabelled environment. Sound classification and automatic speech recognition are difficult areas for SSL to see its success, but it has already been successful in domains such as natural language processing and computer vision.

Large annotated datasets are a major training component of traditional ESC techniques, especially when deep learning models are used. This pipeline uses semi-supervised learning (SSL) for ESC and Urban sounds, which is yet largely unexplored in this area. To overcome the drawbacks of conventional training frameworks, our approach makes use of the intrinsic structure of the data to achieve meaningful results without the need for a lot of manual annotation.

A new methodology is introduced based on the use of label ontology for semi-supervised learning, which allows the implementation of coarse-to-fine learning in the case of ESC. Our approach leverages label ontology, a hierarchical framework that characterizes labels according to their semantic linkages, to drive the semi-supervised learning process.

Using label ontology, we intend to provide efficient hierarchical representation learning in which the environmental sounds can capture both high-level semantic information and low-level acoustic features. Unlike conventional supervised learning approaches that require huge annotated datasets, our approach utilizes all the unlabeled environmental sound data at our disposal. The proposed method of using label ontology and semi-supervised learning is meant to bypass the shortcomings of supervised learning especially when labeled data is challenging to obtain due to lengthy or costly processes.

The approach relies on the principle of semi-supervised learning and uses coarse-to-fine learning on the label ontology of prominent sound classification datasets, such as ESC-50, ESC-10, and UrbanSound8K. These data provide a wide diversity of recorded environmental sounds, including multiple sound types, such as bird noises, urban sounds, and environmental sounds. Both ESC-50 and ESC-10 have been used to support evaluation tasks, where ESC-50 has 2000 recordings from 50 classes while ESC-10 is the subset, having 500 recordings that are divided over 10 classes lowering the complexity of the evaluations while

maintaining diversity. UrbanSound8K consists of 8,732 audio excerpts collected into 10 categories of urban sounds, capturing a vast variety of urban sounds and actions. Using these datasets, we have created a system of sound labels that would enable our model to learn general sound categories first, to then fine-tune its understanding of more specific sounds for better classification.

Our contributions, to this study, can be summarized in the following way:

1. We are proposing a novel approach that combines the use of label ontology alongside semi-supervised learning (SmSL) for ESC. This approach seeks to address the limitations associated with conventional supervised learning models.
2. The implementation of our ideas utilizes label ontology, a hierarchical framework that is based on semantic linkages of sound labels to allow for effective coarse-to-fine learning.
3. We are comparing our proposed method with the conventional supervised learning approaches and other self-supervised learning strategies with experimentation and evaluation on datasets such as ESC-50, ESC-10, and UrbanSound8K. The results indicate that the proposed method performs better than the others.

Problem

Challenges in Current ESC Methods

Environmental Sound Classification (ESC) faces unique challenges primarily due to the unstructured nature of environmental sound data, the low signal-to-noise ratio (SNR) typically encountered, and the diversity of acoustic environments. Traditional supervised learning methods, while effective, heavily depend on large, well-annotated datasets which are costly and time-consuming to prepare. These methods struggle to adapt when data is scarce or noisy, as often found in real-world settings.

Moreover, the reliance on manual feature extraction methods such as MFCC or Wavelet features imposes additional computational burdens and limits the adaptability of the systems to new, unseen types of sounds. While deep learning approaches offer improvements by automating feature extraction and leveraging complex network architectures, they still primarily rely on vast amounts of labeled data, which is not always feasible to obtain.

Limitations of Semi-Supervised and Self-Supervised Approaches

Semi-supervised and self-supervised learning strategies have emerged as potential solutions to the data scarcity problem. These methods reduce dependency on labeled data by utilizing large amounts of unlabeled data. However, the effectiveness of these approaches can be inconsistent, as they often require substantial volumes of high-quality unlabeled data. Moreover, current semi-supervised techniques might not adequately capture the complex hierarchical relationships within sound data, which are crucial for high-level semantic understanding and detailed classification.

Solution

Leveraging Label Ontology for Coarse-to-Fine Learning

To address these challenges, we propose a novel semi-supervised learning framework that incorporates a coarse-to-fine learning pipeline, guided by label ontology. This approach is designed to maximize the utilization of available labeled and unlabeled data and to enhance the learning process by introducing a structured, hierarchical learning objective.

- **Label Ontology Framework**: At the core of our approach is the use of label ontology—a systematic hierarchy of sound labels that organizes sounds from general to specific. This ontology serves as the backbone for our learning process, where the model first learns to identify broad categories (coarse labels) before progressing to more detailed distinctions (fine labels).
- **Pretext Task for Coarse Label Prediction**: Utilizing a Large Language Model (LLM), we define a novel pretext task that involves predicting coarse labels based on sound features extracted from the unlabeled data. This task enables the model to learn useful representations without needing detailed label information, thus making efficient use of the unlabeled data.
- **Fine-tuning with Supervised Learning**: After the model has learned to predict coarse labels, it is fine-tuned using a smaller set of labeled data to make accurate predictions at the fine level. This two-stage learning process ensures that the model develops a robust understanding of sound categories, which enhances its ability to classify new, unseen sounds.
- **Integration with Existing Datasets**: The implementation of our framework is tested on established datasets like UrbanSound8K, ESC-10, and ESC-50, which offer a diverse range of environmental sounds. The hierarchical structure of our label ontology is designed to align with the categories represented in these datasets, ensuring that the model can learn effectively across different types of sound environments.
- **Evaluation and Comparison**: To validate our approach, we compare the performance of our model against baseline models trained with traditional supervised and other semi-supervised methods. Our preliminary results show an accuracy improvement of 5% to 12% over these baselines, demonstrating the effectiveness of the coarse-to-fine learning strategy and the utilization of label ontology.

Dataset

We train and report our proposed approach performance on the UrbanSound8K, ESC-50, and ESC-10 datasets. These are benchmark datasets for audio classification tasks.

UrbanSound8K was made from recordings uploaded on FreeSound which makes the overall dataset very diverse with 8732 sound excerpts (¡=4s) from 10 different classes in WAV format. The dataset also provides metadata to distinguish between the sounds and each slice of a sound from the same occurrence.

The Environmental Sound Classification (ESC-50) dataset contains 2000, 5 second long sound recordings of environmental audio and is arranged into 50 classes, each containing 40 samples. The sounds were also extracted from FreeSound. The ESC-10 dataset is a subset of the larger ESC dataset with 10 classes.

We train our pipeline on these datasets using cross-validation on their predefined folds. The model is trained on N folds where one fold is taken for testing and the rest N-1 is used in training and validation in the ratio 9:1. For UrbanSound8K we used the predefined 10 folds and performed 10-fold cross-validation and the predefined 5 folds to perform 5-fold cross-validation on the ESC datasets. Wetrain the model N times until each fold has been taken for testing once.

Label Ontology

We introduce a novelty in our pipeline for audio classification tasks where we utilized Large Language Models (LLMs) to create an ontology of the classes in the dataset. The LLM is prompted to generate a predefined number of parent classes based on the similarities among the dataset's classes.

Given a dataset with a specified number of classes, the objective is to utilize Large Language Models (LLMs) to generate a predefined number of parent classes based on the similarities among the dataset's classes. The following prompt template illustrates the approach:

```
Dataset Classes: \{classes\}

Number of Classes in Dataset = \{N\}
Number of Parent Classes to Generate = \{P\}

The task is to create P parent classes by identifying and
leveraging the similarities across the \{N\} classes in the
dataset.
```

The table below describes the label ontology for N = 10 classes in UrbanSound8K and ESC-10 with different parent classes ranging from P = 2 to P = 5. The LLM converts the

10 classes into these parent classes which are the used in training a pretext task where the model classifies P classes instead of N.

	UrbanSound8K	ESC-10
$p = 2$	Human and Animal Activities Mechanical and Environmental Noises	Natural Sounds Human and Man-Made Sounds
$p = 3(\sqrt{n})$	Machinery and Construction Sounds Emergency and Alerts Urban and Street Life	Nature and Animals Mechanical Sounds Human-Related Sounds
$p = 5$	Construction and Maintenance Transportation and Vehicles Emergency and Alerts Urban Leisure Residential and Animal Sounds	Natural Elements Mechanical and Man-made Sounds Human-Generated Sounds Animal Sounds Water-related Sounds

Methodology and Code

This section details the motivation behind the problem and the proposed method.

Data Preprocessing

Data preprocessing plays a pivotal role in preparing environmental sound data for our analysis. This subsection delineates the steps undertaken to preprocess the audio data from the ESC-50, ESC10, and UrbanSound8K datasets.

Dataset Partitioning

The datasets are organized into predefined folds, which are utilized to systematically partition the dataset into training, validation, and testing sets. For each experiment, one specific fold is allocated for testing, while the remaining folds are employed for training and validation in the ratio 9:1.

We now create the label ontology as described in \ref{ontology} and set the number of classes in the dataset as P.

```
class Dataset(Dataset):
    def __init__(self, dataframe, fold=None, val=False, test=False):
        self.fold = fold
        all_folds = [i for i in range(1, 11)]
        test_fold = 10-fold
        all_folds.remove(test_fold)
        if test==False:
            df = dataframe[dataframe['fold'].isin(all_folds)]
            train_df, val_df = train_test_split(df, test_size=0.1, random_state=42)
            if val ==True:
                self.dataframe = val_df
            else:
                self.dataframe = train_df

        elif test==True:
            self.dataframe = dataframe[dataframe['fold'] == test_fold]
```

Signal Processing

The audio signals are converted from raw audio files into Mel Spectrograms. The audio is loaded at a fixed sampling rate of 16 kHz. Subsequently, we utilize Librosa's functionality to the Log-Mel Spectrograms which are then resized to 224x224. Additionally,

to adapt the spectrograms for use with models pre-trained on RGB images, we replicate the spectrogram across three channels, simulating a three-channel image format.

```
    def __getitem__(self, index):
        path_to_file = self.get_path_to_file(index)
        signal = self.preprocess_signal(path_to_file)

        x = np.stack([cv2.resize(signal, (224, 224)) for _ in range(3)])

        y = self.dataframe.classID.values[index]
        return torch.tensor(x, dtype=torch.float), y

    def get_path_to_file(self, index):
        return f'urbansound8k/fold{self.dataframe.fold.values[index]}/{self.dataframe.slice_file_name.values[index]}'
    def preprocess_signal(self, path_to_file):
        signal, _ = librosa.load(path_to_file, sr=16000)
        signal = librosa.feature.melspectrogram(y=signal)
        signal = librosa.power_to_db(signal)
        return signal
```

Pretext Task for Coarse-Learning

The pretext task plays a crucial role in our methodology by leveraging the concept of transfer learning to enhance the model's ability to generalize from parent classes, established in the Dataset section.

```
trainPath = 'urbansound8k/'
trainData = pd.read_csv(f'{trainPath}/UrbanSound8K.csv')
trainData.head()
```

	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing

```
pretext=False
if pretext:
    for i in range(len(trainData)):
        value = trainData['class'][i]
        if value == "jackhammer" or value == "engine_idling" or value == "gun_shot" or value == "drilling":
            trainData.loc[i, 'classID'] = 0
        if value == "air_conditioner" or value == 'street_music' or value == 'siren' or value == "car_horn":
            trainData.loc[i, 'classID'] = 1
        if value == 'children_playing' or value == 'dog_bark':
            trainData.loc[i, 'classID'] = 2
```

Initially, the architectures are trained exclusively on the parent classes generated by the LLMs. This training phase aims to capture high-level features and similarities among the dataset's classes, encoded within the parent classes. By focusing on these broader categories, the models learn to recognize overarching patterns that are intrinsic to groups of similar classes, rather than the finer details specific to each class.

Upon completion of the training on parent classes, the weights of the CNN are saved and the classifier head is discarded. These weights encapsulate the learned representations of

the parent classes, serving as a valuable pre-trained foundation. The strategy here is to exploit these pre-trained weights to accelerate and enhance the learning process when the model is subsequently fine-tuned on the detailed class structure of the dataset.

Fine-tuning the Pretext

After the initial training phase on parent classes, the next critical step in our methodology is fine-tuning the model on the entire dataset using the weights of the CNN from the pretext. This fine-tuning phase is designed to specialize the model's knowledge, enabling it to distinguish between the more detailed and specific classes in the dataset. This section outlines the process and considerations involved in fine-tuning the encoders during the pretext task.

Loading Pre-trained Weights

The fine-tuning process begins by loading the architectures with the weights saved from the pretext task. These weights, which were obtained by training on parent classes, contain generalized features and patterns relevant to the dataset. By starting with these pre-trained weights, we leverage the knowledge already acquired, aiming to build upon this foundation to achieve more nuanced class differentiation.

```
def get_pretext_loaded(model, pretext_path):
    backbone_weights = torch.load(pretext_path, map_location='cpu')
    model_dict = dict(model.resnet50.state_dict())
    #print(backbone_weights["model_state"].keys())
    for key, weights in backbone_weights["model_state"].items():
        a = key[9:]
        if a.find('fc') == -1: #key should not contain fc
            model_dict[a] = weights #update all keys except fc

    model.resnet50.load_state_dict(model_dict)
    a = model_dict['conv1.weight'][0]
    b = backbone_weights["model_state"]["resnet50.conv1.weight"][0]

    if np.array_equal(np.array(a), np.array(b)):
        print("Backbone Loaded")
    else:
        print("Backbone not loaded")
    return model
```

New Classifier Head

Consistent with the baseline training approach, a new classifier head is attached to the encoder. While the underlying architecture remains the same, this new head replaces the one used during the pretext task, reflecting the shift from broad parent classes to the detailed class structure inherent to the dataset.

```python
class CustomResNet50(nn.Module):
    def __init__(self, num_classes=10):
        super(CustomResNet50, self).__init__()
        # Load a pre-trained ResNet50
        self.resnet50 = models.resnet18(pretrained=True)
        # Freeze all layers in ResNet50
        for param in self.resnet50.parameters():
            param.requires_grad = True  # Set true to unfreeze

        # Get the input features of the original fully connected layer
        in_features = self.resnet50.fc.in_features

        # Replace the fully connected layer with custom layers
        self.resnet50.fc = nn.Sequential(
            nn.Linear(in_features, 512),
            nn.ReLU(),
            #nn.Dropout(0.2),
            nn.Linear(512, 256),
            nn.ReLU(),
            #nn.Dropout(0.2),
            nn.Linear(256, num_classes),
        )

        # Unfreeze the custom fully connected layers
        for param in self.resnet50.fc.parameters():
            param.requires_grad = True

    def forward(self, x):
        return self.resnet50(x)
```

Fine-tuning Strategy

The fine-tuning strategy involves training the entire model — both the pre-loaded architecture (or encoder) and the new classifier head — on the dataset. The process is carefully managed to prevent overfitting, with hyperparameters such as learning rate and batch size finely adjusted to balance learning new features against preserving the generalized knowledge obtained during the pretext task.

The training loop:

1. Set all hyperparameters and initialize the Dataloader and Model and set the number of folds for the model to train.

```
from time import time
warnings.filterwarnings("ignore")


start_time = time()
folds = 10
for fold in range(7, folds):
    checkpoint_path = f'resnet18_baseline_us8k_fold{fold+1}.pth'
    prextext_path = f'resnet18_pretext_urbansound8k_3classes_fold1.pth'
    batch_size = 32
    trainSet = Dataset(trainData, fold=fold)
    valSet = Dataset(trainData, fold=fold, val=True)
    testSet = Dataset(trainData, fold=fold, test=True)
    trainLoader = DataLoader(trainSet, batch_size=batch_size, shuffle=True, num_workers = 2)
    valLoader = DataLoader(valSet , batch_size=batch_size, num_workers = 2)
    testLoader = DataLoader(testSet , batch_size=batch_size, num_workers = 2)

    print('Training set: {}, Validation set: {}, Test Set: {}'.format(len(trainSet), len(valSet), len(testSet)))
    print("Folds of training set:", set(list(trainSet.dataframe["fold"].values)))
    print("Folds of validation set:", set(list(valSet.dataframe["fold"].values)))
    print("Folds of test set:", set(list(testSet.dataframe["fold"].values)))
    print("Classfying", num_classes, "classes.")

    model = CustomResNet50(num_classes=num_classes)

    if torch.cuda.device_count() >= 2:
        print(f"Using {torch.cuda.device_count()} GPUs!")
        device = torch.device("cuda")
        model = torch.nn.DataParallel(model)  # Wrap the model for multi-GPU use
        model.to(device)
    else:
        device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
        model.to(device)
        print("Using single GPU or CPU")

    epochs = 50
    optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)
    cost = torch.nn.CrossEntropyLoss()
    best_val_accuracy = 0.0
    best_val_loss = 10000
    checkpoint_path = checkpoint_path
```

2. Implement a normal supervised classification RFOR loop where the training and validation accuracy are calculated. The loss is calculated and updated through the optimizer.

```
for epoch in range(start_epoch, epochs):
    train_loss = 0
    val_loss = 0
    train_correct = 0
    val_correct = 0

    model.train()
    for x, y in tqdm(trainLoader):
        optimizer.zero_grad()
        x,y = x.to(device),y.to(device)
        pred = model(x)
        loss = cost(pred, y)
        train_loss += cost(pred, y).item()
        train_correct += (pred.argmax(1) == y).type(torch.float).sum().item()
        loss.backward()
        optimizer.step()

    model.eval()
    with torch.no_grad():
        for x, y in tqdm(valLoader):
            x,y = x.to(device),y.to(device)
            pred = model(x)
            loss = cost(pred, y)
            val_loss += cost(pred, y).item()
            val_correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    train_loss = train_loss/len(trainLoader)
    val_loss = val_loss/len(valLoader)
    train_accuracy = train_correct / len(trainData)
    val_accuracy = val_correct / len(valSet)
```

3. Load and test the trained model with the highest validation accuracy on each fold and note the accuracies.

```
checkpoint_path=checkpoint_path
checkpoint = torch.load(checkpoint_path)
model.load_state_dict(checkpoint['model_state'])
model.eval()
test_correct = 0
for x, y in tqdm(testLoader):
    x,y = x.to(device),y.to(device)
    pred = model(x)
    pred = torch.softmax(pred, axis=1)
    test_correct += (pred.argmax(1) == y).type(torch.float).sum().item()
test_accuracy = test_correct / len(testSet)
print(f"Test accuracy for fold: {fold+1} is: {test_accuracy*100}")
#change test accuracy in dataframe where fold = fold and epoch is max
df.loc[(df.Fold == fold) & (df.Epoch == df[df.Fold == fold].Epoch.max()), "Test Acc"] = test_accuracy
df.to_csv("training_metadata.csv", index=False)
```

4. Average the accuracies from each fold for the final aggregate test accuracy of the model on the dataset.

UrbanSound8K	**Baseline**		**Proposed - 3 classes**	
Test Fold	Frozen	Unfrozen	Frozen	Unfrozen
10		87.6		80.04
9		83.7		87.4
8		77.9		79.5
7		82.1		88.1
6		80.07		81.7
5		83.4		89
4		83.8		87.1
3		70.7		82.5
2		80.5		85.4
1		75.9		82.8
Average		80.567		84.354

Results & Discussion

We evaluate the performance of our pipeline on the classification task of these datasets. We utilize 4 state-of-the-art architectures in classification - ResNet18, ResNet50, EfficientNetB0 and EfficientNetB1.

Implementation details

We use the PyTorch framework for experimentation and model development. All training and testing are carried out on an NVIDIA GPU P100. Every model is taken in a complete log-mel spectrogram of size 223*224*3 and is stacked 3 times, to create an RGB channel. The batch size was set to 32 and Adam was used as the optimizer in the training. The learning rate was initialized to 0.0001, trained on 50 epochs and saved based on the best validation accuracy.

Loss Function: Since we are working with a multi-classification problem, we experimented with standard Cross-Entropy Loss.

$$H(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y_{ij}})$$

Where $H(y, \widehat{y})$ is the Cross-Entropy Function, y_{ij} is a binary indicator of whether class j is the correct classification for observation i. $\widehat{y_{ij}}$ is the predicted probability that observation i is of class j. N is the number of samples and C is the number of classes.

Hyperparameters

The training of the classification architecture were configured with a set of hyperparameters optimized for sound classification.

A batch size of 32 was chosen to balance the trade-off between memory usage and the stability of the gradient estimates. We employed the Adam optimizer for its adaptive learning rate capabilities, setting an initial learning rate of 0.0001. This choice enhances the convergence towards optimal performance.

The loss function selected was Cross-Entropy, which is well-suited for multi-class classification problems. The training was conducted over 50 epochs, and the model was saved when it achieved a higher validation accuracy.

Comparison with previous works

SUMMARY OF MODEL PERFORMANCES ON ESC-10

Model	Accuracy (%)	Classifier
PiczakCNN [18]	80.50	CNN
GoogleNet [1]	63.20	CNN
Tokoz et al. [23]	91.30	EnevNet
Zhang et al. [27]	91.70	CNN
Triplet loss [9]	90.28	Residual Network
Temporal GAP [21]	90.53	Residual Network
Tripathi et al. [24]	91.70	SSL + Residual Network
ECHO	96.00	SmSL + Residual Network
ECHO	97.50	SmSL + CNN

As shown in the above Table, we compare previous state-of-the-art models with the proposed method, with a focus on the work by Tripathi et al., which achieved a notable accuracy of 91.70% and serves as a benchmark in our evaluation. The authors employed a self-supervised learning strategy, which involved training on data-augmented spectrogram signals through a pretext task.

After training, the pretext model is frozen and subsequently finetuned for specific dataset classes. Our proposed technique, detailed in the Methodology Section, significantly surpasses this benchmark by delivering a 4.3% improvement in accuracy using semi-supervised learning with the residual networks and a 5.8% improvement in accuracy using semi-supervised learning with CNNs.

Comparison between the Accuracies of Baseline and Proposed

ACCURACIES OF BASELINE AND PROPOSED

Model	ESC-50		ESC-10		US8K	
	Baseline	Proposed	Baseline	Proposed	Baseline	Proposed
ResNet50	76.35	**84.20**	91.25	**96.00**	80.57	**84.35**
ResNet18	80.75	**81.75**	98.25	95.75	80.02	**84.22**
EfficientNetB0	77.55	**82.50**	91.25	**94.50**	82.36	**86.97**
EfficientNetB1	80.10	**86.05**	98.50	97.50	80.93	**87.66**

We utilized the ResNet18, ResNet50, EfficientNetB0, and EfficientNetB1 architectures leveraging their pre-trained weights on the ImageNet dataset. We fine-tune these models through a straightforward classification strategy, where the final linear layer is adjusted to match the number of distinct classes within our dataset.

As delineated in the above table, these configurations are designated as our Baseline models, for which we present their classification accuracies. Subsequently, we introduce and evaluate the performance of our proposed pipeline. The results demonstrate that our approach significantly outperforms the baseline models, with improvements in accuracy ranging from **3%** to nearly **8%**.

Ablation Study

ABLATION STUDY RESULTS ACROSS DIFFERENT DATASETS

ESC-50			ESC-10			US8K		
3	5	$7(\sqrt{n})$	2	$3(\sqrt{n})$	5	2	$3(\sqrt{n})$	5
81.55	81.45	**84.2**	94.00	94.50	**96.00**	84.17	**84.35**	82.40

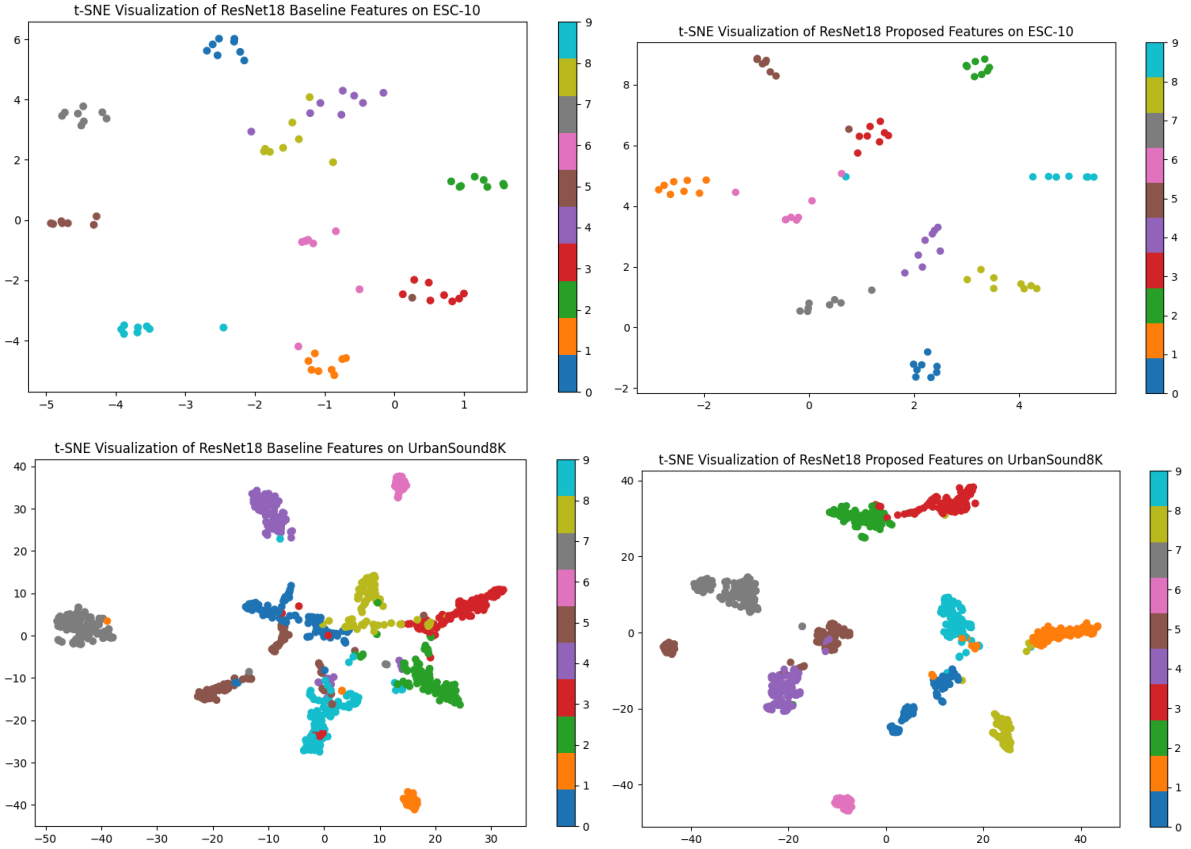
We have performed training across 3 datasets i.e. UrbanSound8K, ESC10 and ESC50 on the ResNet50 architecture. Here the sub categories taken are the number of classes taken for Pretext Training in the Label Ontology set as P. On performing cross validation and measuring accuracy, we recorded appreciable numbers for almost all categories.

Amongst all the categories, the ones with $P = \sqrt{n}$ in ESC-50 and UrbanSound8K were recorded with the highest accuracy while in ESC-10 the highest accuracy was recorded with $P = 5$. This is due to the minimal size of the dataset making it very easy for the model to learn the features.

t-SNE Graphs

The feature embeddings from the trained model are viewed in a t-distributed Stochastic Neighbor Embedding(t-SNE) graph. Using this unsupervised learning technique we view the reduced embeddings from the output of the second last layer in ResNet18 where the embeddings are 256-dimensional.

The embeddings are viewed for datasets ESC-10 and UrbanSound8K by testing on the last predefined fold of each dataset. The graphs are provided below. The graphs show more accurate and precise clusters formed through are proposed methodology compared to conventional finetuning classification. The points in the baseline graphs are further apart compared to the proposed graphs where different clusters are further apart and points from the same clusters are closer to each other.



Limitations

The approach we have discussed is not a novel architecture but a pipeline based on our results in the table shown in Comparison with Previous Works.

Our approach can be further implemented for state-of-the-art models where we predict a significant increase in accuracy.

Moreover, the experiments were only conducted with multi-classification where the prediction results in only one label based on the Softmax probabilities. In the future, we plan to propose multi-label classification on benchmark datasets like AudioSet and FSD50K using our approach for better generalization.

Conclusion

We provide a novel method for environmental sound classification that employs semi-supervision to learn from coarse-to-fine level on the label ontology of the state-of-the-art sound datasets such as ESC-50, ESC-10, and UrbanSound8K. Through providing the tools for granular soundscape environment, our approach has shown the achievement of better accuracy with improvement rates from 5% up to 12% compared to the baseline models. With the utilization of the structural hierarchies in sound labels, our method can be attained. This eases the learning process and learns meanings for both high-level semantic information and low-level acoustic features. Experimentation across wide range of supervised learning approaches in audio classification show our architecture outperforms these and highlights the importance of using the label ontology in self-supervised learning for environmental sound classification.

References

Abdul, Z. K., & Al-Talabani, A. K. (2022). Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, *10*, 122136–122158.

Boddapati, V., Petef, A., Rasmusson, J., & Lundberg, L. (12 2017). Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, *112*, 2048–2056.

Chi, P.-H., Chung, P.-H., Wu, T.-H., Hsieh, C.-C., Chen, Y.-H., Li, S.-W., & Lee, H.-Y. (2021). Audio ALBERT: A Lite BERT for Self-supervised Learning of Audio Representation. In *arXiv [eess.AS]*. http://arxiv.org/abs/2005.08575

Chu, S., Narayanan, S., & Kuo, C.-C. J. (2009). Environmental sound recognition with time--frequency audio features. *IEEE Trans. Audio Speech Lang. Processing*, *17*(6), 1142–1158.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Dieleman, S., & Schrauwen, B. (2014). End-to-end learning for music audio. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6964–6968.

Gong, Y., Chung, Y.-A., & Glass, J. R. (2021). AST: Audio Spectrogram Transformer. *CoRR*, *abs/2104.01778*.

Grollmisch, S., & Cano, E. (2021). Improving Semi-Supervised Learning for Audio Classification with FixMatch. *Electronics*.

Gururani, S., & Lerch, A. (2021a). *Semi-supervised audio classification with partially labeled data*.

Gururani, S., & Lerch, A. (2021b). Semi-Supervised Audio Classification with Partially Labeled Data. In *arXiv [cs.SD]*. http://arxiv.org/abs/2111.12761

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 770–778.

Jansen, A., Plakal, M., Pandya, R., Ellis, D. P. W., Hershey, S., Liu, J., Moore, R. C., & Saurous, R. A. (2018). Unsupervised Learning of Semantic Audio Representations. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 126–130.

Ji, C., Mudiyanselage, T. B., Gao, Y., & Pan, Y. (2021). A review of infant cry analysis and classification. *EURASIP Journal on Audio, Speech, and Music Processing*, *2021*.

Kipf, T. N., & Welling, M. (2016). *Semi-supervised classification with graph convolutional networks*.

Kumaran, U., RadhaRamMohan, S., Nagarajan, S. M., & Prathik, A. (2021). Fusion of mel and gammatone frequency cepstral coefficients for speech emotion recognition using deep C-RNN. *International Journal of Speech Technology*, *24*, 303–314.

Lee, J., Park, J., Kim, K. L., & Nam, J. (2018). SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification. *Applied Sciences*, *8*, 150.

Lin, C.-C., Chen, S.-H., Truong, T.-K., & Chang, Y. (2005). Audio classification and categorization based on wavelets and support vector Machine. *IEEE Transactions on Speech and Audio Processing*, *13*, 644–651.

Liu, A. T., Li, S.-W., & Lee, H.-Y. (2021). TERA: Self-Supervised Learning of Transformer Encoder Representation for Speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 2351–2366.

Lu, L., Zhang, H., & Li, S. Z. (2003). Content-based audio classification and segmentation by using support vector machines. *Multimedia Systems*, *8*, 482–492.

McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. *Proceedings of the 14th Python in Science Conference*.

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., & Ogata, T. (2014). Audio-visual speech recognition using deep learning. *Applied Intelligence*, *42*, 722–737.

O'Shea, K., & Nash, R. (n.d.). *An Introduction to Convolutional Neural Networks*. Arxiv.org. Retrieved April 7, 2024, from http://arxiv.org/abs/1511.08458

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. *Proceedings of the 23rd ACM International Conference on Multimedia*.

Salamon, J., Jacoby, C., & Bello, J. P. (2014). A dataset and taxonomy for urban sound research. *Proceedings of the 22nd ACM International Conference on Multimedia*.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., & Raffel, C. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *arXiv [cs.LG]*. http://arxiv.org/abs/2001.07685

Tagliasacchi, M., Gfeller, B., Quitry, F., & Roblek, D. (04 2020). Pre-Training Audio Representations With Self-Supervision. *IEEE Signal Processing Letters*, *PP*, 1–1.

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional Neural Networks. *ICML*, *abs/1905.11946*.

Tarvainen, A., & Valpola, H. (2018). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *arXiv [cs.NE]*. http://arxiv.org/abs/1703.01780

Tokozume, Y., & Harada, T. (2017). Learning environmental sounds with end-to-end convolutional neural network. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Tokozume, Y., Ushiku, Y., & Harada, T. (n.d.). *Between-class learning for image classification*. Arxiv.org. Retrieved April 7, 2024, from http://arxiv.org/abs/1711.10284

Tripathi, A. M., & Mishra, A. (2021). Self-supervised learning for Environmental Sound Classification. *Appl. Acoust.*, *182*(108183), 108183.

Xie, J., & Zhu, M. (2019). Handcrafted features and late fusion with deep learning for bird sound classification. *Ecological Informatics*, *52*, 74–81.

Xu, Y., Kong, Q., Wang, W., & Plumbley, M. D. (2017). Large-Scale Weakly Supervised Audio Classification Using Gated Convolutional Neural Network. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 121–125.

Yang, L., & Zhao, H. (2021). Sound classification based on multihead attention and support vector machine. *Math. Probl. Eng.*, *2021*, 1–11.

Zhang, Zhichao, Xu, S., Cao, S., & Zhang, S. (n.d.). *Deep convolutional neural network with mixup for environmental sound classification*. Arxiv.org. Retrieved April 7, 2024, from http://arxiv.org/abs/1808.08405

Zhang, Zixing, & Schuller, B. (2012). Semi-supervised learning helps in sound event classification. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 333–336.