




Présentation PLDAC **AUDIO**

QI Zhirui
ZHU Zhihao



Index

- Introduction
- Prétraitement
- Analyse des données
- Construction de CNN
- Entraînement et Validation
- Test avancé et tableau de précision
- Conclusion

Introduction

Objectif de projet:

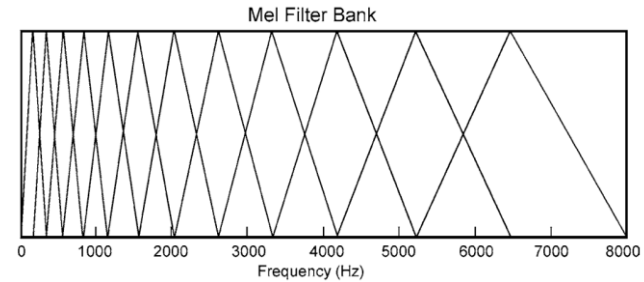
Classification de scènes audio (3 classes: Indoor, Outdoor ou Transport)

(Classes précises: Airport, Subway, Station...)

Source des données des audios:

dataset en DCASE (dans différentes villes)

Technique principale N°1



LMFB(Log-mel filter bank):

Il sert à changer la forme de data par FFT(Fast Fourier Transform) pour faciliter notre chargé de donnée

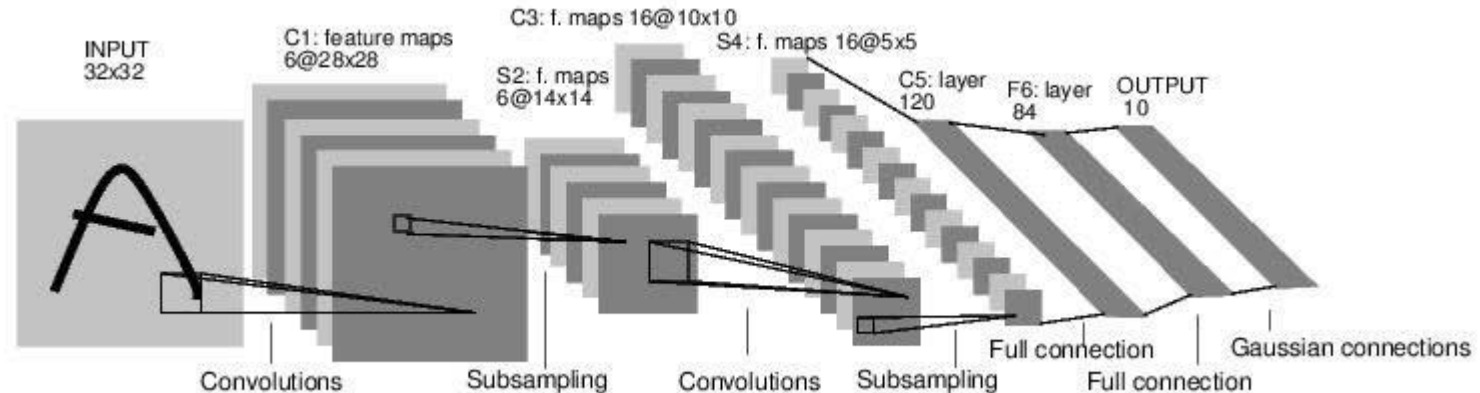
Méthode en général:

1. Transformée de Fourier un signal
2. Utilisez la fenêtre de chevauchement triangulaire pour mapper le spectre à l'échelle de Mel
3. Prendre le Logarithme
4. Prendre une transformation cosinus discrète
5. MFCC est le spectre converti.

Technique principale N°2

CNN(convolutional neural network):

réseau de neurones à réaction directe, ses neurones artificiels peuvent répondre à une partie des unités environnantes dans la zone de couverture, a d'excellentes performances pour le traitement d'image à grande échelle.



Prétraitement

- lire tous les fichiers
- traiter par la méthode `get_logmel_data`
- enregistrer dans une matrice de dimension (nombre d'échantillon, 1, 128, 431)

Algorithm 1: `get_logmel_data`

Entrée:

`filepath` : string

Paramètre :

- `duration`=10
- `num_freq_bin`=128
- `num_fft`=2048
- `num_channel`=1

Sortie :

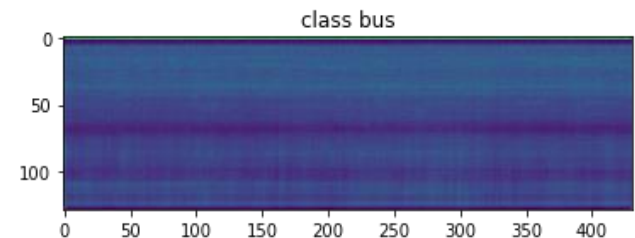
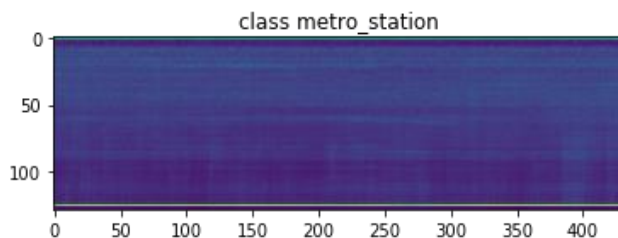
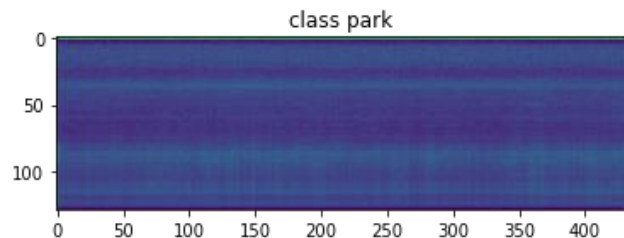
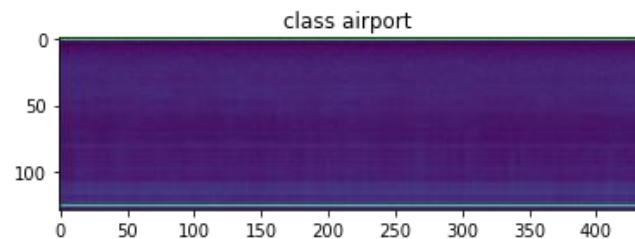
`logmel_data` : matrices de données de dimension (1,128,431)

- 1 Lire le fichier de *filepath*
 - 2 `hop_length` est défini par $\left\lfloor \frac{\text{num_fft}}{2} \right\rfloor$
 - 3 `num_time_bin` est défini par $\left\lceil \frac{\text{duration} \times sr}{\text{hop_length}} \right\rceil$ avec *sr* est taux d'échantillonnage
 - 4 *logmel_data* est initialisé par 0 de dimension (*num_channel*, *num_freq_bin*, *num_time_bin*).
 - 5 On calcule la valeur mel avec paramètre qu'on a défini avant.
 - 6 On calcule le logarithme de la valeur mel
-

Analyse des données

- airport : la somme de variance = 235197.98
- bus : la somme de variance = 270929.2
- metro station : la somme de variance = 347128.25
- park : la somme de variance = 269944.06

--- la variance



Analyse des données

--- la similarité

modèle cosinus

- classe airport vs classe bus : $1.3630869e-06$
- classe airport vs classe metro station : $1.3439948e-06$
- classe airport vs classe park : $1.0636244e-06$
- classe bus vs classe metro station : $1.2277217e-06$
- classe bus vs classe park : $9.777448e-07$
- classe metro station vs classe park : $9.379438e-07$

Construction de CNN

Network N°1:

la structure est de

2 couches cachées convolutional

2 couches de MAX pooling

3 couches linéaires ReLU

input module

Conv(1,6,3) - ReLU

Max pooling(2, 2)

Conv(6,16,3) - ReLU

Max pooling(2, 2)

Linear(16*3180, 120) - ReLU

Linear(200, 84) - ReLU

Linear(84, *nb_classe*)

Network N°2:

la structure est de

2 couches cachées convolutional

2 couches de MAX pooling

3 couches linéaires ReLU

input module

Conv(1,6,3) - ReLU

Max pooling(2, 2)

Conv(6,18,3) - ReLU

Max pooling(2, 2)

Linear(18*3180, 120) - ReLU

Linear(200, 84) - ReLU

Linear(84, *nb_classe*)

Network Compliqué:

la structure est de

10 couches cachées convolutional

3 couches de MAX pooling

10 couches Batchnormalisation ReLU

1 couche de AVG pooling

Drop out, Softmax, Gaussien Noise..

Trop grand pas réalisable pour notre ordinateur, un peu triste

input module
Conv(1,42,5)(pad-2,stride-2) Batchnormalisation(42) - ReLU Conv(42,42,3)(pad-1,stride-1) Batchnormalisation(42) - ReLU Max pooling(2, 2) + GaussianNoise(1.00)
Conv(42,84,3)(pad-1,stride-1) Batchnormalisation(84) - ReLU Conv(84,84,3)(pad-1,stride-1) Batchnormalisation(84) - ReLU Max pooling(2, 2) + GaussianNoise(0.75)
Conv(84,168,3)(pad-1,stride-1) Batchnormalisation(168) - ReLU Drop out(0.3) Conv(168,168,3)(pad-1,stride-1) Batchnormalisation(168) - ReLU Drop out(0.3) Conv(168,168,3)(pad-1,stride-1) Batchnormalisation(168) - ReLU Drop out(0.3) Max pooling(2, 2) + GaussianNoise(0.75)
Conv(168,336,3)(pad-1,stride-1) Batchnormalisation(336) - ReLU Drop out(0.5) Conv(336,336,1)(pad-1,stride-1) Batchnormalisation(336) - ReLU Drop out(0.5)
Conv(336,10,1)(pad-1,stride-1) Batchnormalisation(10) GaussianNoise(0.3) Global-Average-Pooling
10-way Soft-Max

Entraînement et Validation

Algorithm 2: Train_visualisation

Entrée:

- net : Réseau
- optimizer : type optim pour défini la méthode de la descente de gradient
- train_x : matrice taille (nombre d'échantillon train, nombre de channel , dimension x de donnee , dimension y de donnee)
- train_y : liste taille (nombre d'échantillon train, 1)
- val_x : matrice taille (nombre d'échantillon validation, nombre de channel , dimension x de donnee , dimension y de donnee)
- val_y : vliste taille (nombre d'échantillon validation, 1)
- paramètre :**
 - lossfonction=nn.CrossEntropyLoss() : fonction du cout
 - pourcentage_validation=0.9 : pourcentage de validation (utilise si *val_x* ou *val_y* est vide)
 - nbite=1 : nombre d'époch de backward
 - batch_size=1 : taille de batch pour chaque itération
 - visual=True : decide si on affiche le graphe du cout

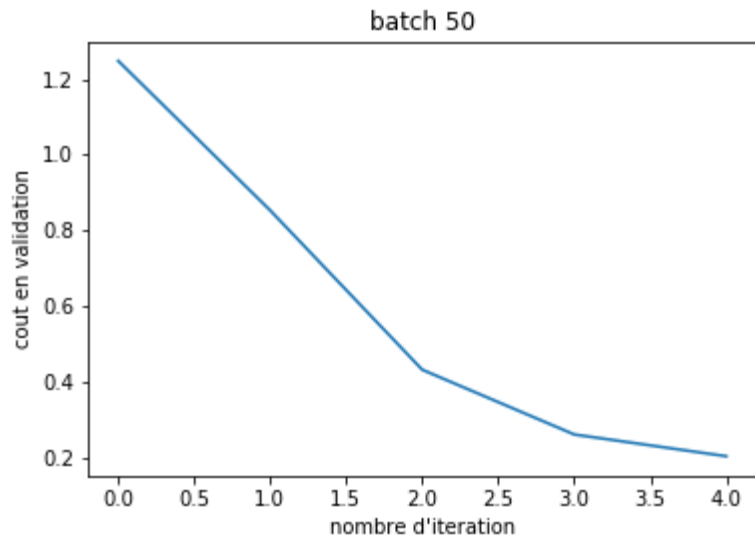
Sortie :

matrices_confusion (dimension (nb_classe, nb_classe)), net

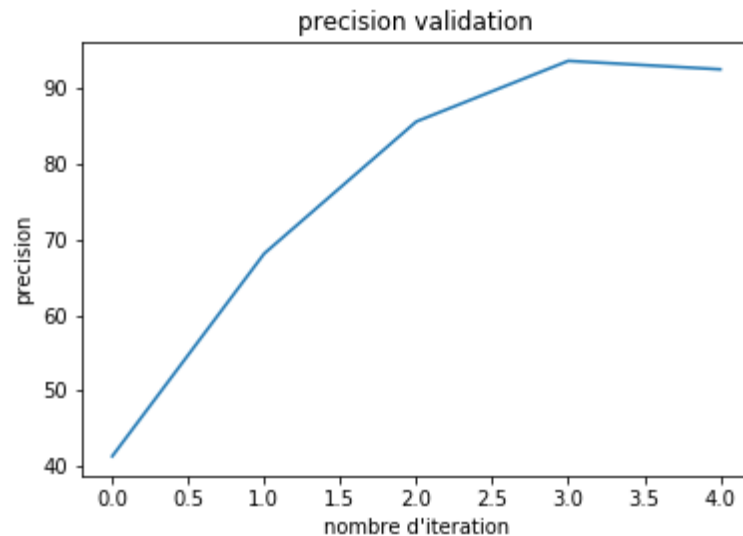
Entraînement et Validation

```
1 if val_x ou val_y est vide then
2   |   Separation de validation;
3 end
4 Tester si batch_size est plus grand que la taille de donnee;
5 Confirmer les echantillons de train sont de ordre arbitraire;
6 for epoch  $\in$  nbite do
7   |   for i, data  $\in$  enumerate(dataset) do
8     |   Extraire dataTrain et label;
9     |   Mettre le paramètre gradients en zéro;
10    |   forward + backward + optimize;
11    |   Calculer le coût de validataion pour chaque renouvellement de pararmète
12  end
13  Calculer la precision de validataion pour chaque epoque;
14  Calculer le coût de validataion pour chaque epoque;
15 end
16 visualiser et affichage;
```

Exemple d'exécution : net 1 : 4 époque avec 50 mini-batch

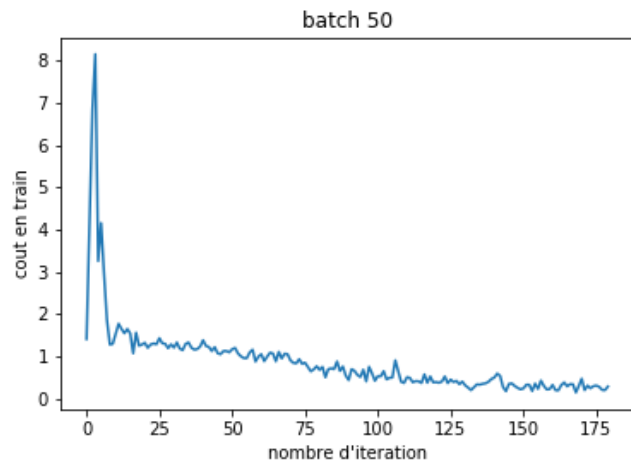


coût d'exécution chaque époque

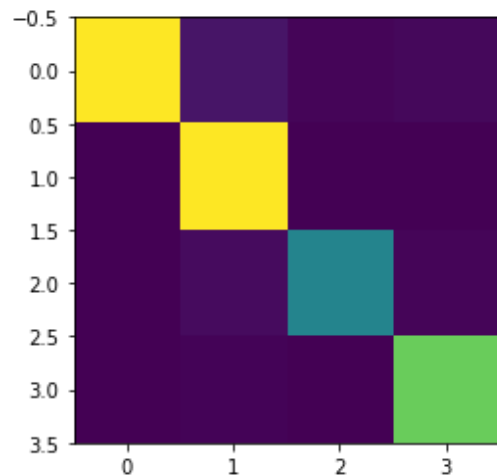


precision validation chaque époque

Exemple d'exécution : net 1 : 4 époque avec 50 mini-batch



coût d'exécution chaque itération



matrice confusion

533	30	8
0	12	
0	534	1
1	1	
1	15	
0	240	7
0	5	0
	413	

le réseau a tendance à confondre :

- bus et metro station
- airport et bus
- airport et metro station

Test avancé et tableau de précision

Net domaines	<i>Net 1</i>	<i>Net 2</i>
Cas : 90% validation, et 5 epoch avec 50 mini-batch		
pourcentage de la validation	90%	90%
loss	0.13427214324474335	0.1327710896730423
precision validation	95.55555555555556 %	96.33333333333334 %
Precision de Reseau	96%	89 %
Precision de airport	100%	100 %
Precision de Bus	100%	100 %
Precision de metro station	84%	64 %
Precision de park	100%	93 %
Cas : 90% validation, et 50 epoch avec batch(1800)		
pourcentage de la validation	90%	90%
loss	0.47955480217933655	0.3835340738296509
precision validation	88.5%	90.94444444444446 %
Precision de Reseau	53%	51 %
Precision de airport	70%	72 %
Precision de Bus	8%	32 %
Precision de metro station	62%	32 %
Precision de park	72%	68 %

Conclusion

En résumé, on utilise 3 CNN créés pour traiter notre donnée et on a enfin les performances sont environ 90%. D'ailleurs, on recherche sur la relation entre les coûts et nombre d'itération et batch. On sait ainsi l'importance de prétraitement des données comme LMFB, par conséquent, ça facilite notre étude suivante.

Après tout, comme on a indiqué, c'est la première fois qu'on utilise CNN pour traiter le problème et ça nous pose intéressant et enrichissant à la fin.

Merci Beaucoup!