# CP-JKU SUBMISSIONS TO DCASE'19: ACOUSTIC SCENE CLASSIFICATION AND AUDIO TAGGING WITH RECEPTIVE-FIELD-REGULARIZED CNNS

Technical Report

*Khaled Koutini*[1*], *Hamid Eghbal-zadeh*[1,2†], *Gerhard Widmer*[1,2]

[1]Institute of Computational Perception (CP-JKU) & [2]LIT Artificial Intelligence Lab,
Johannes Kepler University Linz, Austria
khaled.koutini@jku.at

## ABSTRACT

In this report, we detail the *CP-JKU* submissions to the DCASE-2019 challenge Task 1 (acoustic scene classification) and Task 2 (audio tagging with noisy labels and minimal supervision). In all of our submissions, we use fully convolutional deep neural networks architectures that are regularized with Receptive Field (RF) adjustments. We adjust the RF of variants of Resnet and Densenet architectures to best fit the various audio processing tasks that use the spectrogram features as input. Additionally, we propose novel CNN layers such as *Frequency-Aware CNNs*, and new noise compensation techniques such as *Adaptive Weighting for Learning from Noisy Labels* to cope with the complexities of each task. We prepared *all* of our submissions *without the use of any external data*. Our focus in this year's submissions is to provide the best-performing *single-model* submission, using our proposed approaches.

***Index Terms—*** Acoustic Scene Classification, audio tagging, noisy labels, CNNs, Receptive Field Regularization

## 1. INTRODUCTION

In this year's DCASE-2019 challenge, we target three tasks: Acoustic Scene Classification (ASC) without mismatch (Task 1.A), ASC under mismatched condition (Task 1.B) [1], and Audio Tagging with noisy labels (Task 2) [2]. We focus our efforts on providing variants of successful architectures that can achieve good performance with a single model. We base all of our submissions on a recent study of ours [3], where we analyzed the effect of Receptive Field (RF) tuning on the performance of various architectures in ASC. We modify the RF of various architectures such as Resnet [4], Shake-shake [5], and Densenet [6] according to the guidelines provided in [3]. An analysis of the RF of a model and its effect on the model's performance is provided in Section 3. In addition, we propose a new idea that we call *Frequency-Aware Convolutional Neural Networks (CNNs)*, where the frequency locations for the input are added as an additional channel to a CNN layer. This new approach is detailed in Section 3.1.3.

In each task, we take advantage of recent advances in machine learning to adjust our models to the task at hand. To deal with the distribution mismatch, we incorporate the newly proposed Shake-shake architecture [5], a Resnet variant that is known to generalize better. Additionally, we use Maximum Mean Discrepancy (MMD) [7], a kernel method that can be used for domain adaptation and transfer learning in Deep Neural Networks (DNNs).

To overcome the complexities of learning from noisy labels in Task 2, we explore various new ideas. We propose an *adaptive weighting* approach (detailed in Section 5.3), a sample selection strategy for noisy labels (detailed in Section 5.4), and a class weighting based on the *label weighted label ranking average precision (lwlwrap)* (detailed in Section 5.5) to adapt the way our model learns from the noisy labels.

## 2. EXPERIMENTAL SETUP

### 2.1. Data Preparation

We extracted the input features using a Short Time Fourier Transform (STFT) with a window size of 2048 and 25% overlap. We perceptually weight the resulting spectrograms and apply a Mel-scaled filter bank in a similar fashion to Dorfer et al. [8]. This preprocessing results in 256 Mel frequency bins.

#### *2.1.1. Task 1*

The input is first down-sampled to 22.05 kHz for Task 1. We use mono signal for Task 1.B. We process each channel independently for Task 1.A and provide the CNN with a two-channel-spectrogram input. The input frames are normalized using the training set mean and standard deviation.

#### *2.1.2. Task 2*

For Task 2, we keep the original sampling rate of 44.1 kHz due to the wide range of classes, We suspect that some information encoded in higher frequencies can be useful to distinguish some of the classes. The input frames are then normalized using the curated training set mean and standard deviation.

### 2.2. Optimization

We used Adam [9] with a specific scheduler. We start training with a learning rate of $1 \times 10^{-4}$. From epoch 50 until 250, the learning rate decays linearly from $1 \times 10^{-4}$ to $5 \times 10^{-6}$. We train for another 100 epochs with the minimum learning rate $5 \times 10^{-6}$ in a setup similar to [3]. In the case where there is no validation set (e.g. single model submissions), we average the predictions of the model every 5 epochs starting from epoch 300, in order to reduce the variance caused by picking the model at a specific epoch.

*Responsible for Task 1.A and Task 2
†Responsible for Task 1.B

### 2.3. Data Augmentation

We use *mix-up* [10] since it was shown to have a great impact on the performance and the generalization of the models.

### 2.4. Model Averaging

**Stochastic Weight Averaging:** Stochastic Weight Averaging (SWA) [11] had led to a better performance on the validation set in our experiments on Task 2. We keep an SWA copy of the model parameters while training. This SWA average is updated with new parameters every 3 epochs.
**Snapshot Averaging**: A snapshot of the model during training is saved every 5 epochs, within the last 100 epochs. The predictions of all these models are then averaged for the final prediction. The snapshot averaging is computationally more efficient than training separate models, and easier than stochastic weight averaging approaches such as [11] for creating ensemble models, as no re-computation of the layer statistics (such as batch-norm) is required.
**CV Averaging**: A model is trained on different CV folds, and the predictions of models from different folds are averaged. This approach provides more robustness, as different models have seen different training data points, hence found slightly different minima. This approach is simple yet effective, and was used successfully in our previous submissions [12, 8, 13].

## 3. ARCHITECTURES

We adapted ResNet and DenseNet variants following the guidelines of our previous work [3]. Using the provided development set for Task 1.A, we performed a grid search on the RF of the ResNet architecture. We concluded that the optimal RF for the Task 1.A dataset is around $90 \times 90$ pixels of our extracted spectrograms (Section 2.1). Figure 1 shows the validation loss of the provided development set, for ResNet with different RFs using the mono input. The steps taken to produce this search will be published at `https://github.com/kkoutini/cpjku_dcase19`.

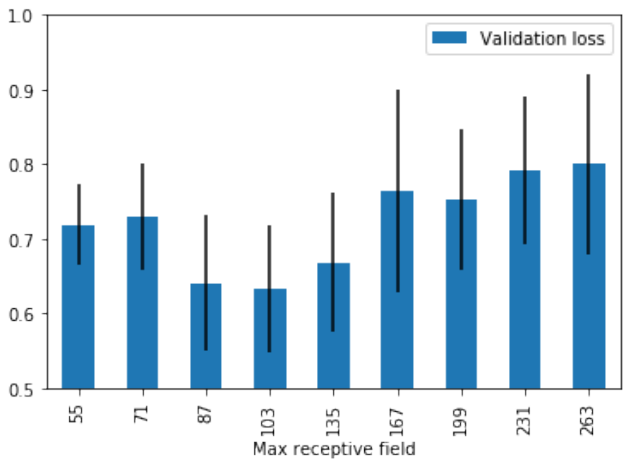


Figure 1: Validation Loss of the provided development split of Task 1 a dataset, for ResNet with different receptive fields over a mono input.

In Task 1.B, we used network architectures with the same receptive field as in Task 1.A, motivated by the similarity and overlap between the two datasets, as they are both about modeling the acoustic scenes. For Task 2, we searched for the optimal RF using a 4 fold Cross-Validation (CV) of the curated data only. We found that the optimal receptive field for the dataset is around $100 \times 100$ pixels over the extracted spectrograms.

### 3.1. ResNet

ResNet [4] and its variants (such as preact-ResNet [14]) achieve state-of-the-art results in image recognition. As we showed in our recent study [3], such architectures can be adapted to audio tasks using RF regularization. We first adapt the RF of the ResNet as explained above. The resulting network architectures are detailed in Table 1. Both RN1 and RN2 have the same RF. Note that we used RN1 in Task 1.B since it has fewer layers which allows for a larger batch size as explained in Section 4.2.

We used different variants of ResNet, with the following modifications.

#### *3.1.1. Shake-Shake Regularization*

The Shake-Shake architecture [5] is a variant of ResNet that is proposed for improved stability and robustness. Each residual block has 3 branches, which are summed with random coefficients (in both the forward and backward pass) [5].

#### *3.1.2. Pre-act ResNet*

Pre-act ResNet is a ResNet variant where residual branches are summed up before applying the non-linearity [14].

#### *3.1.3. Frequency-aware CNNs*

Since we are using fully convolutional networks, learned filters are agnostic to the frequency information of the feature maps. In other words, the spectrograms and feature maps can be rolled over both the time and frequency dimension with a minor impact on the network predictions. We propose a novel convolutional layer, namely the *Frequency-aware Convolution*, to make filters more specialized in certain frequencies by concatenating a new channel containing the frequency information[1] of each spatial pixel to each feature map. The CNN models that incorporate our frequency-aware layer will be called the *Frequency-Aware Convolutional Neural Networks (FACNNs)* in our report. We denote the value of the pixel with spatial index $(f, t)$ in the new channel as $V(f, t)$; it is calculated as follows:

$$V(f, t) = f/F \tag{1}$$

where $F$ is the size of the frequency dimension of the feature map, $f$ is the pixel index in the frequency dimension, and $t$ is the pixel index in the time dimension. This new channel gives the convolutional filters a frequency context.

### 3.2. DenseNet

We adapted DenseNet [6] to DN1, in a similar fashion to our study in [3]. The resulting network has a maximum receptive field of around $90 \times 90$ pixels.

[1] In this report, we used a number between 0 and 1, where 0 represents the lowest frequency, and 1 represents the highest frequency in the spectrogram. But this range can be adapted according to the value range of the input.

Table 1: Modified ResNet architectures

| RB Number | RB Config | |
|---|---|---|
| | ***RN1*** | ***RN2*** |
| | Input $5 \times 5$ stride=2 | |
| 1 | $3 \times 3$, $1 \times 1$, P | $3 \times 3$, $1 \times 1$, P |
| 2 | $3 \times 3$, $3 \times 3$, P | $3 \times 3$, $3 \times 3$, P |
| 3 | $3 \times 3$, $3 \times 3$, | $3 \times 3$, $3 \times 3$ |
| 4 | | $3 \times 3$, $1 \times 1$, P |
| 5 | $3 \times 3$, $1 \times 1$, P | $1 \times 1$, $1 \times 1$ |
| 6 | | $1 \times 1$, $1 \times 1$ |
| 7 | | $1 \times 1$, $1 \times 1$ |
| 8 | | $1 \times 1$, $1 \times 1$ |
| 9 | $1 \times 1$, $1 \times 1$ | $1 \times 1$, $1 \times 1$ |
| 10 | | $1 \times 1$, $1 \times 1$ |
| 11 | | $1 \times 1$, $1 \times 1$ |
| 12 | | $1 \times 1$, $1 \times 1$ |

RB: Residual Block, P: $2 \times 2$ max pooling after the block.
RB number 1-4 have 128 channels.
RB number 5-8 have 256 channels.
RB number 9-12 have 512 channels.

## 4. ACOUSTIC SCENE CLASSIFICATION (TASK 1)

### 4.1. ASC without mismatch (Task 1.A)

**Single model submission:** Our first submission consists of the predictions of a single model trained on the whole development set with no validation set. We average the prediction of the model every 5 epochs, after epoch 300. The results on the Kaggle leader boards are provided in Table 2.
**Average over 4-fold cross-validation submission:** We randomly split the development set into 4 folds regardless of the city or the scene id. We then train 4 models on these splits and submitted the average of the predictions of these models.
**Average different variants submissions:** We trained different variants of DenseNet and ResNet (as explained in Section 3) and averaged different combinations of these models for our submissions 3 and 4 (16 models and 7 models respectively).

### 4.2. ASC with mismatched recording devices (Task 1.B)

Task 1.B is to classify audio snippets into acoustic scene classes, where the model has to deal with the nuisance of a recording device mismatch between the training and the testing data. A small set of parallel recordings are provided in the training to be used as adaptation data, however, the majority of the training data is from a single device which is not targeted in the challenge test set. Hence, a model has to learn device-invariant representations such that the model can generalize on the new samples from various devices.

As studied in [15], CNNs have shown to suffer from significant performance degradation when dealing with mismatched distributions at inference time. To tackle this issue, in our submission to Task 2.B we integrate a *Two-Sample Test (TST)* [2] based on kernel Maximum Mean Discrepancy (MMD) [7] to encourage our model to learn device-invariant embeddings, such that the embedding distribution of various devices are indistinguishable from each other.

One advantage of TST frameworks compared to pair-wise matching techniques is that, as these methods are trying to match the distributions, they do not require paired recordings. This provides more freedom in terms of mini-batch selection and data augmentation, as the model only needs two sets of (even unpaired) samples from two devices.

Another family of models capable of dealing with unpaired distribution matching are adversarial frameworks such as Generative Adversarial Networks (GANs) [16]. One advantage of the TST frameworks over the adversarial ones is that TST frameworks do not require a parametric discriminator. Training a discriminator increases the complexity of the model, and introduces new problems to the training such as instability and mode collapse, which require additional efforts to solve [17].

#### *4.2.1. The Proposed Domain Adaptation Objective*

We begin with the definition of our MMD measure between two sets of samples from two different distributions. Let $k$ be the Gaussian RBF kernel. Then we use an estimation of $\text{MMD}(P, Q)$ between two distributions $P$ and $Q$, given a set of $m$ samples $X = \{X_1, \ldots, X_m\} \overset{iid}{\sim} P$ and a set of $m$ samples $Y = \{Y_1, \ldots, Y_m\} \overset{iid}{\sim} Q$,[3] defined as:

$$\begin{aligned}\text{MMD}(X, Y) := &\frac{1}{m^2} \sum_{i \neq i'} k(X_i, X_{i'}) + \frac{1}{m^2} \sum_{j \neq j'} k(Y_j, Y_{j'}) \\ &- \frac{2}{m^2} \sum_{i \neq j} k(X_i, Y_j)\end{aligned} \tag{2}$$

As explained in [7], $\text{MMD}_k(P, Q) = 0$ if and only if $P = Q$.

We now define $\mathcal{F}_l(x)$ as the activations of a hidden layer $l$, in a feed-forward neural network $\mathcal{F}$, given an input spectrogram $x$. Given the definitions above, our domain adaptation loss to learn device-invariant representations in layer $l$ is as follows:

$$\mathcal{L}_{\text{MMD}} = \text{MMD}(\mathcal{F}_l(x_i^d)_{i=1}^m, \mathcal{F}_l(x_j^{d'})_{j=1}^m) \tag{3}$$

where $x_i^d$ and $x_j^{d'}$ are two spectrograms of audio snippets from devices $d$ and $d'$, respectively. Please note that $x_i^d$ and $x_j^{d'}$ are not required to be parallel recordings of the same scene, hence are sampled randomly in every minibatch.

For classification, we use Categorical Cross Entropy (CCE) loss, which is combined with our domain adaptation loss as follows:

$$\mathcal{L}_{\text{HYB}} = \mathcal{L}_{\text{MMD}} + \mathcal{L}_{\text{CCE}} \tag{4}$$

where $\mathcal{L}_{\text{HYB}}$ is our hybrid loss, and $\mathcal{L}_{\text{CCE}}$ is the categorical cross entropy loss used for classification.

#### *4.2.2. Training setup*

Our training consists of two steps: 1) the classification step, and 2) the hybrid step. In the classification step, we train our model for a full epoch with the CCE loss, using a randomly selected mini-batch of size 10 from the full training set. Each mini-batch is augmented with *mix-up* [10]. In the hybrid step, our model is trained using the

[2] Comparing samples from two probability distributions, by proposing statistical tests of the null hypothesis that these distributions are equal against the alternative hypothesis that these distributions are different [7].

[3] We assume the number of samples from the two distributions are equal.

hybrid loss, on a randomly-selected set of parallel samples[4]. First, two devices are randomly chosen out of the 3 available devices, then a number of 10 samples from each selected device is randomly selected to create a 20-samples mini-batch. This mini-batch is further augmented via *mix-up*. Our model is trained iteratively using the classification and the hybrid steps.

**Architectures**: We use two receptive-field-regularized CNN architectures from [3], namely Resnet and Shake-Shake [5]. We use RN1 (Section 3.1) with Shake-Shake and our proposed Frequency-Aware CNN layers. The exact architecture of our ResNet is provided in Table 1, but with half the number of channels. We use a 4-fold Cross-Validation (CV), and use the average prediction of the resulting 4 models in our final prediction. Additionally, we use snap-shot averaging as explained above.

#### *4.2.3. Results*

In the following, we detail our 4 submissions to the DCASE'19 Task 1.B. The results on the Kaggle leader-boards are provided in Table 2.
**1) Shake-Shake/ResNet with Snapshot averaging:** In this submission, we use our ResNet and Shake-Shake model. From each architecture, 4 models are trained using our 4 folds CV. The predictions from models of all the 4 folds are averaged for each test file. Additionally, we use the average prediction of 20 snapshots from each model as explained above for our final prediction.
**2) Shake-Shake/ResNet:** Similar to our first submission, this submission uses an average prediction of our ResNet and Shake-Shake architectures. Instead of snap-shot averaging, only the best model from each fold was used in the final prediction averaging.
**3) Shake-Shake with Snapshot averaging:** This submission is similar to our first submission, but only Shake-Shake model is used.
**4) Shake-Shake:** In this submission, a similar approach to our second submission is used, but only using our Shake-Shake model.

## 5. AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION (TASK 2)

### 5.1. Baseline

We trained a DenseNet DN1 (Section 3.2) and different variants of RN1 (Section 3.1) on the curated data. The best performing ResNet-based model on our cross-validation was models with both Shake-Shake and the Frequency-Aware CNNs.

### 5.2. Low Learning-rate Noisy Training

To integrate the noisy data, we trained our models on the noisy labels but with a small learning rate that is 0.2 of the learning rate used for training on the curated data.

### 5.3. Adaptive-Weighting of Noisy Samples

We calculate the *inner product* of the noisy ground truth vector with the vector of the predicted probabilities from the noisy samples. We then average the resulting scalar while training each sample. The resulting average is then used to weight the loss of each noisy sample, when seen by the network in the future epochs. This will weight down the noisy samples, based on the ground truth and network predictions disagreement.

[4]Samples from the training set that have recordings in all A, B and C devices.

| | SID | Kag. Pub | Kag. Prv. |
|---|---|---|---|
| Task 1.A | | | |
| Single Res Snp | 1 | 82.83 | 81.33 |
| CV Res Snp | 2 | 83.50 | 83.00 |
| CV Res-FACNN Snp | - | 80.66 | 83.66 |
| Preact\Res-FACNN Snp | 3 | 80.50 | 82.50 |
| Preact\Res-FACNN\Dense Snp | 4 | 82.00 | 82.83 |
| Task 1.B | | | |
| CV Res\Shake-FACNN Snp | 1 | 75.00 | 77.50 |
| CV Res\Shake-FACNN | 2 | 74.33 | 76.83 |
| CV Shake-FACNN Snp | 3 | 75.66 | 75.16 |
| CV Shake-FACNN | 4 | 74.33 | 75.83 |
| CV Shake-FACNN (no DA) | - | 73.66 | 73.00 |
| Task 2 | | | |
| Average 1 | 1 | .728 | - |
| Average 2 | 2 | .725 | - |
| CV Shake-FACNN | - | .715 | - |

Table 2: The accuracy of different methods on various test sets. Snp: Snapshot averaging, Shake: Shake-Shake ResNet, Res: ResNet. DA: Domain Adaptation. The details of the submissions are explained in their respective sections with their corresponding Submission Ids (SID).

### 5.4. Noisy Samples Selection

At a later stage, we average the weights of the noisy samples (as explained in the previous section) that resulted from different runs. We use a threshold to select only samples with higher weights, to be used for training new models.

### 5.5. Inverse-Lwlwrap Class Weighting

We noticed that some classes are harder to learn, or have lower Lwlwrap scores. To tackle this issue, we use the inverse of the lwlwrap of a class, to weight the class's vector term in the cross-entropy loss function.

### 5.6. Submissions

We use the average of different combinations of the models trained with the aforementioned setups for our final submissions. The results on the Kaggle leader-board are provided in Table 2.

## 6. CONCLUSION

In this technical report, we detailed our approaches to tackle Task 1 (A and B) and Task 2 of the DCASE-2019 challenge. We showed that our RF regularized CNNs can achieved high ranks on the Kaggle public leaderboards in various tasks. We introduced several novel techniques such as FACNNs and Adaptive-weighting, and demonstrated their effectiveness on various DCASE-2019 datasets. This suggests that these architectures can be an effective tool that offer good generalization properties for various audio processing tasks .

## 7. ACKNOWLEDGMENT

This work has been supported by the COMET-K2 Center of the Linz Center of Mechatronics (LCM) funded by the Austrian federal government and the federal state of Upper Austria.

## 8. REFERENCES


[1] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13. [Online]. Available: https://arxiv.org/abs/1807.09840

[2] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," *arXiv preprint arXiv:1906.02975*, 2019.

[3] K. Koutini, H. Eghbal-zadeh, M. Dorfer, and G. Widmer, "The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[5] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[7] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[8] M. Dorfer, B. Lehner, H. Eghbal-zadeh, C. Heindl, F. Paischer, and G. Widmer, "Acoustic Scene Classification with Fully Convolutional Neural Networks and I-Vectors," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Challenge (DCASE2018)*, 2018.

[9] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond Empirical Risk Minimization," 2017. [Online]. Available: http://arxiv.org/abs/1710.09412

[11] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[12] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," in *DCASE 2016-challenge on Detection and Classification of Acoustic Scenes and Events*. DCASE2016 Challenge, 2016.

[13] B. Lehner, H. Eghbal-Zadeh, M. Dorfer, F. Korzeniowski, K. Koutini, and G. Widmer, "Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task," in *DCASE 2017-challenge on Detection and Classification of Acoustic Scenes and Events*. DCASE2017 Challenge, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *arXiv preprint arXiv:1603.05027*, 2016.

[15] H. Eghbal-zadeh, M. Dorfer, and G. Widmer, "Deep within-class covariance analysis for robust deep audio representation learning," in *Neural Information Processing Systems, Interpretability and Robustness in Audio, Speech, and Language Workshop*, 2018.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[17] H. Eghbal-zadeh, W. Zellinger, and G. Widmer, "Mixture density generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.