Report - Mixture Density Networks

Antoine Debouchage[1,2], Valentin Denée[1], and Clement Wang[1,2]

[1]Ecole Normale Supérieure Paris Saclay
[2]CentraleSupélec

November 15, 2024

Abstract

In this report, we delve into the implementation and comprehensive evaluation of Mixture Density Networks (MDNs), inspired by the seminal work of Christopher M. Bishop in 1994 [1]. Our study focuses on the reimplementation of Bishop's original MDN architecture, exploring its efficacy across diverse datasets. The experimentation encompasses two distinct toy datasets, where the MDN is applied for inverse function prediction, showcasing its versatility in capturing non-injective relationships. Furthermore, we extend our analysis to the realm of robot kinematics, investigating the MDN's ability to model intricate robotic motion patterns. To gauge the model's adaptability to real-world challenges, we conduct experiments on the MNIST dataset for digit prediction. The results not only highlight the MDN's performance in capturing intricate patterns but also shed light on its potential for probabilistic predictions, crucial in scenarios with inherent uncertainty.

The report starts by explaining Mixture Density Networks (MDNs) and the idea behind mixture models. We then dive into how we implemented MDNs based on Bishop's 1994 paper, covering the architecture and training process for making probabilistic predictions. Next, we get practical with experiments. First up is a simple sinusoid toy dataset, where MDNs show their knack for predicting inverse sinusoidal patterns. We then take MDNs to the world of robot kinematics, using them to model the two-mode motion of an extended robotic arm. Lastly, we put MDNs to the test on the MNIST dataset [3], demonstrating their ability to predict digits with uncertainty. Through these hands-on experiments, we aim to show how MDNs can be versatile and effective across different types of data and applications.

Our code is openly accessible at this link `https://github.com/clementw168/mixture-density-net`.

1 Mixture Model

1.1 Theoretical foundation

Define p the probability density of the target data as a linear combination of kernel functions:

$$p(t|x) = \sum_{i=1}^{m} \alpha_i(x)\phi_i(t|x) \tag{1}$$

where m is the number of components in the mixture. The parameters $\alpha_i(x)$ are called *mixing coefficients* and can be regarded as prior probabilities (conditioned on x) of the target vector t having been generated from the i^{th} component of the mixture. Note that the mixing coefficients are taken to be functions of the input vector x. The functions $\phi_i(t|x)$ represent the conditional density of the target vector t for the i^{th} kernel. Various choices for the kernel functions are possible. Here, as in the paper, we use kernel functions which are Gaussian of the form:

$$\phi_i(t|x) = \frac{1}{(2\pi)^{c/2}\sigma_i(x)^c} exp\left(-\frac{||t-\mu_i(x)||^2}{2\sigma_i(x)^2}\right) \tag{2}$$

where the vector $\mu_i(x)$ represents the centre of the i^{th} kernel, with components μ_{ik} ,c is the size of vector t. We have also assumed that the components of the output vector are statistically independent within each component of the distribution, and can be described by a common variance $\sigma_i(x)$.

1.2 Comparison with conventional MSE

Define the sum-of-squares error, defined over a set of data, and which has to be minimized in the usual approach to network training:

$$E^S(w) = \frac{1}{2}\sum_{k=1}^{c} \iint (f_k(x;w) - t_k)^2 p(t,x)\, dt\, dx$$

where t_k represent the components of the target vector, $f_k(x; w)$ denote the corresponding outputs of the network mapping function, which is parametrized by an array w (the weights and biases), and p is a joint probability density. We can minimize the error function in:

$$f_k(x, w^*) = \langle t_k \mid x \rangle$$

where w^* represents the corresponding set of weight values, and where $\langle Q \mid x \rangle = \int Q(t)p(t|x)\,dt$ is the conditional average of the quantity $Q(t)$. Thus, the network function is given by the conditional average of the target data, conditioned on the input vector.

Most conventional applications of neural networks only make use of the prediction for the mean, given by $f_k(x; w)$, which approximates the conditional average of the target data, conditioned on the input vector. For classification problems, this represents the optimal solution. However, for problems involving the prediction of continuous variables, the conditional average represents only a very limited statistic. For many applications, there is considerable benefit in obtaining a much more complete description of the probability distribution of the target data. That is the main interest of introducing the Mixture Density Network which can in principle represent arbitrary conditional distributions, in the same way that a conventional neural network can represent arbitrary non-linear functions.

2 IMPLEMENTATION

2.1 Mixture density network

We aim to determine the parameters α_i and $\phi_i(t|x)$ for a set of m models using neural networks. Introducing a Gaussian prior to $\phi_i(t|x)$ simplifies the process of extracting μ_i, σ_i, and α_i values through neural network computations. Consequently, for an output space of dimension k, our network architecture involves mapping inputs x to a vector space of dimension $(k + 2) \times m$.

We call *Mixture Density Network* (MDN) the combined structure of a feed-forward network and a mixture model. By choosing a mixture model with a sufficient number of kernel functions, and a neural network with a sufficient number of hidden units, the MDN can approximate as closely as desired any conditional density $p(t|x)$. Here is an illustration of an MDN structure.

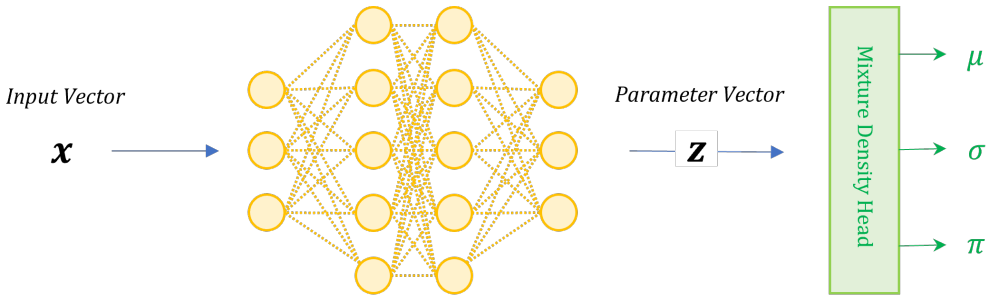


Figure 1: Mixture Density Network architecture as a neural network fed to a mixture model head which will generate the given parameters of the conditional probability density $p(t \mid x)$

To clarify further, each of the m Gaussian models within this architecture generates a vector comprising k values for the mean, a single scalar representing the standard deviation σ_i of the Gaussian distribution, and another scalar indicating the prior weight α_i. This structure enables the neural network to output parameters tailored to each model, facilitating a comprehensive representation of the Gaussian priors on $\phi_i(t|x)$ within the network's output space.

Regarding the output activation functions of the neural network, μ_i is a direct output from a linear function. Meanwhile, for α_i, a softmax function is applied:

$$\alpha_i = \frac{\exp(a_i)}{\sum_{k=1}^{m} \exp(a_k)}$$

Here, a_k denotes the direct output of the neural network without undergoing any additional activation function. So, in particular, we have:

$$\sum_{i=1}^{m} \alpha_i = 1$$

Therefore, the mixing coefficients can be seen as a probability distribution. Employing the softmax function ensures that each α_i assumes a value within the range of 0 to 1, representing relative probabilities,

and importantly, guarantees that the sum of all α_i values equals 1. Lastly, an exponential activation is used to compute the σ_i:

$$\sigma_i = \exp(s_i)$$

This specific activation function constrains σ_i to exclusively positive values and centers them around 1.

It's important to highlight that the input x can take various forms, allowing the MDN structure to function as a versatile regression head applicable to diverse regression tasks, regardless of the input data format. Section 3 of this work demonstrates this adaptability by presenting results involving 1D datasets, images, and audio data.

2.2 Loss function

The loss naturally comes from the negative log-likelihood on the output of the network:

$$l(y_{\text{true}}, \alpha, \mu, \sigma) = -\log\left[\sum_{i=1}^{m} \alpha_i \frac{1}{\sqrt{2\pi}^c \sigma_i^c} \exp\left(-\frac{\|y_{\text{true}} - \mu_i\|^2}{2\sigma_i^2}\right)\right]$$

2.3 Predicting with MDN

For a given input vector x, the most likely value for the output vector is given by the maximum of the conditional density $p(t|x)$. Since this density function is represented by a mixture model, the location of its global maximum is a problem in non-linear optimization. Standard techniques existing for solving such problems are computationally costly, because iterative. Therefore, we need to find a faster approach.

If we assume that the component kernels of the density function are not too strongly overlapping, then to a very good approximation the most likely value of t will be given by the center of the highest component. From (1) and (2), we see that the component with the largest central value is given by:

$$\underset{i}{max}\left\{\frac{\alpha_i}{\sigma_i^c}\right\}$$

The corresponding center μ_i represents the most likely output vector. Alternatively, we can also consider the total probability mass associated with each of the mixture components. This is more appropriate for applications involving multi-valued mappings with a finite number of distinct branches, in which we are interested in finding a representative vector corresponding to the most probable branch.

3 EXPERIMENTS

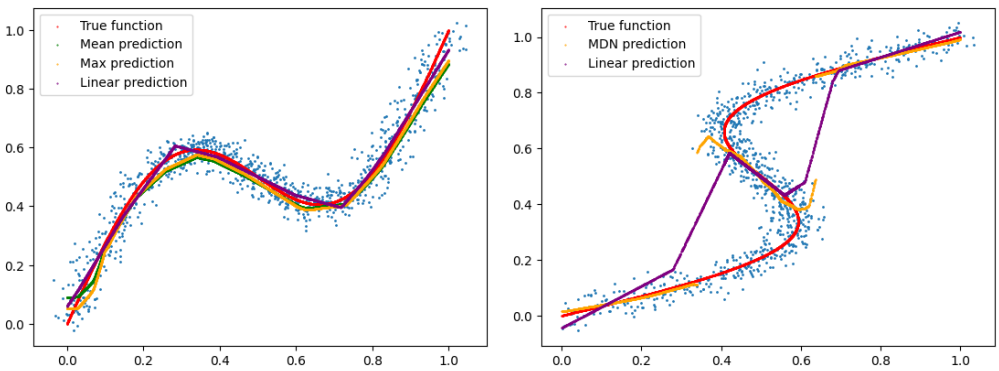
3.1 Sinusoid toy dataset

We started our experiments on a simple toy dataset with the 1D function f:

$$f(t) = t + 0.3\sin(2\pi t)$$

We compared the results of an MDN prediction and a simple Multi-perceptron with $x = t + \epsilon_1$ and $y = f(t) + \epsilon_2$ where ϵ_1 and ϵ_2 represent a Gaussian noise.

Here, we can observe that an MDN gives a similar performance in MSE on a simple toy dataset where there is only one possible output. Next experiment consists in testing with $x = f(t) + \epsilon_1$ and $y = t + \epsilon_2$



(a) Comparison of a simple neural network and a MDN with $m = 3$ on a simple toy dataset

(b) Comparison of a simple neural network and a MDN with $m = 3$ on the hard toy dataset

In this example, a single input x can have multiple acceptable outputs and the average of the multiple acceptable outputs is not necessarily a good output.

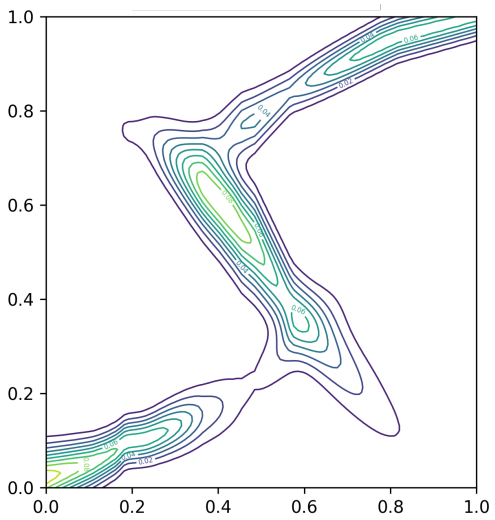


Figure 3: Contour plot of MDN prediction with $m = 3$ on the hard toy dataset

As anticipated, the linear neural network forecasts the average output of the ground truth, whereas the MDN accurately predicts a segment of the correct label.

For deeper exploration into the MDN's effectiveness, we show the α values between two models, one with $m = 3$ and another with $m = 5$. The visual representation (Fig 4) illustrates that in scenarios with a sole possible output, a single component consistently takes precedence. However, when multiple outputs exist, the distribution spreads across several components. Notably, each component aligns with distinct Gaussian centers, a remarkable observation despite not being explicitly enforced within the network structure or the loss function.

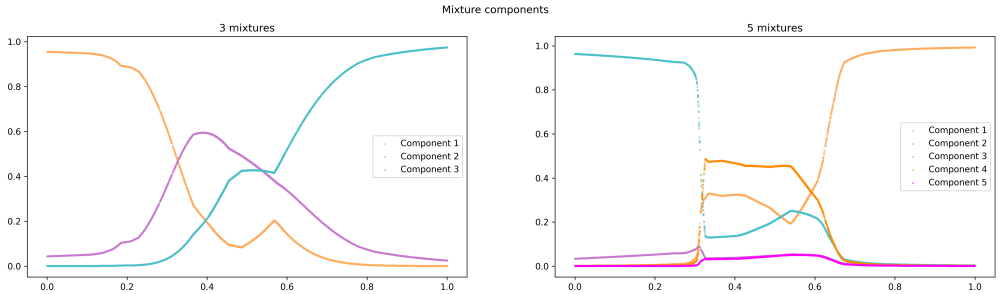


Figure 4: Values of α along the space for each mixture component of a model with $m = 3$ (left) and $m = 5$ (right) on the hard toy dataset.

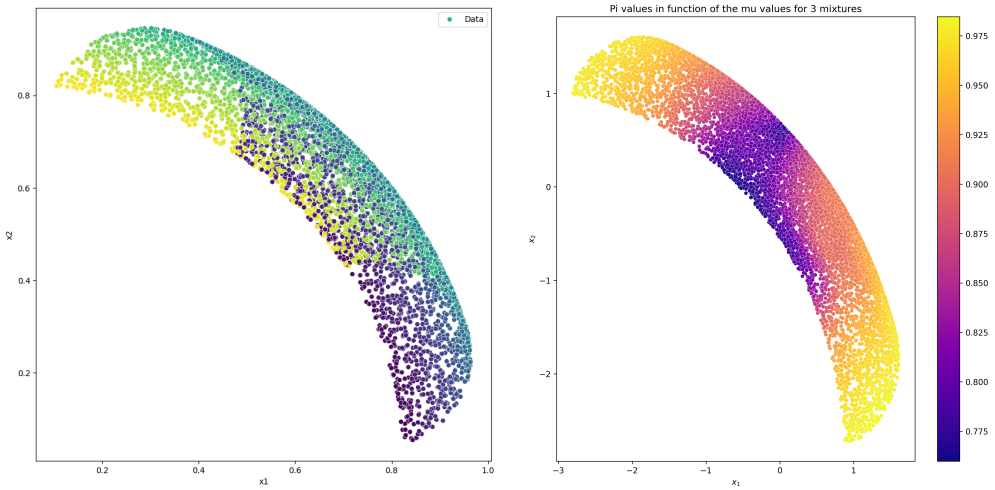
3.2 Robot kinematics

In this section, we delve into another straightforward application of a Mixture Density network: the kinematics of a simple 2-link robot arm. Given values of the joint angles (θ_1, θ_2) the robot arm's end is positioned at (x_1, x_2) according to the following equations:

$$x_1 = L_1 \cos(\theta_1) - L_2 \cos(\theta_1 + \theta_2)$$
$$x_2 = L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2)$$

where (θ_1, θ_2) in range $(0.3, 1.2) \times (\pi/2, 3\pi/2)$, $L_1 = 0.8$ and $L_2 = 0.2$.

The inverse problem of determining (θ_1, θ_2) from (x_1, x_2) is non-injective, as depicted in Figure 5a. The data can be categorized into three areas: both extremities, where the points are injective, and the middle part, which exhibits a double-valued mapping. Therefore, it is particularly well-suited for a Mixture Density Network.



(a) Overlaps of (x_1, x_2) generated by samples of (θ_1, θ_2) in range $(0.3, 1.2) \times (\pi/2, 3\pi/2)$

(b) Best α (also noted π) values for each points predicted

We generate synthetic training (5k points) and testing (10k) datasets using the provided equations. Subsequently, we train both a feed-forward network and a mixture density network with 3 mixtures. To evaluate the performance of both models visually, we calculate the distances between the ground truth positions and the predicted positions across the entire testing dataset. We then draw lines connecting these predicted points to the original ones. The outcome is depicted in Figure 9 in the Appendix. The Mean Square Error of the distances between the predicted and original points gives for the MDN 0.0265 and for the MLP 0.0423.

The Mixture Density Networks demonstrate superiority over traditional feed-forward networks, particularly in the central region with a two-valued mapping. Additionally, as illustrated in Figure 5b, the two extremes are predicted using one mixture with a probability almost equal to one. However, in the two-valued mapping region, the probability decreases to 75%.

3.3 Image regression on MNIST dataset

Demonstrating the practical use of MDNs in image datasets, we employed an MDN training approach on the MNIST dataset [3] to forecast the digit represented by each image. In addressing normalization concerns, we scaled the labels to ensure they fell within the range of 0 to 1.

Both a basic linear neural network and a convolutional neural network were trained to highlight the MDN head's versatility with a CNN. The convolutional MDN model is composed of three convolutional layers, each succeeded by a ReLU activation function and a MaxPooling layer. The architecture follows closely the implementation of the AlexNet [4] architecture without using BatchNormalization layers. Following the convolutional segment, the output traverses two linear layers before reaching the MDN head (Figure 6).

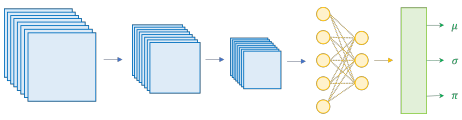
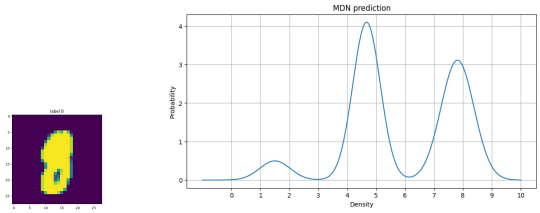


Figure 6: Convolutional Mixture Density Network composed of a standard convolutional network followed by a feed-forward network with the outputs sent into a Mixture Density head.



(a) Sample

(b) Prediction

Figure 7: MDN output on a sample of MNIST

The MDN framework operates akin to a classification head by providing a probability distribution across the network's output. This characteristic facilitates post-processing the output, enabling the derivation of a confidence score and presenting several potential answers.

Figure 7 illustrates it as the decision appears challenging. The MDN surpasses the limitations of a simple linear neural network by offering multiple potential answers. This ability to accommodate uncertainty and present multiple plausible outcomes demonstrates the advantage of MDNs over more straightforward network structures.

4 Discussion and future works

One unresolved challenge lies in determining a suitable metric for complex problems where multiple potential outputs exist. In practical scenarios, acquiring the entire label distribution is uncommon, often limited to having only a single correct label within test samples rather than the complete range of potential labels.

Consider a straightforward instance to illustrate why Mean Squared Error (MSE) falls short as a metric for regressions with multiple possible outcomes. Imagine a fixed 'x' linked to an output distribution: 0.5 for label 0 and 0.5 for label 1. Should we use the argmax of α for regression prediction, the resulting MSE error would be 0.5, whereas a standard MSE regression would yield an error of 0.25. This example highlights the inability to directly compare the Mean Squared Error framework with the Mixture Density Network (MDN) framework using a simple MSE.

A more appropriate metric would entail directly assessing the difference between the actual and predicted distributions, potentially using measurements like the Wasserstein distance or likelihood. However, obtaining the true distribution directly in a straightforward manner is often impractical in real-world scenarios.

Mixture Density Networks offer versatile applications, seamlessly adapting to various scenarios. One such application involves employing them for regression tasks on time series data, particularly in forecasting or audio processing. Our exploration expanded to audio processing over time, as depicted in Figure 11 in the appendix, using the CoughVid dataset [5] to predict either age or susceptibility to infection. Unfortunately, our initial attempts did not yield compelling results. However, in an extended work, one could leverage MDNs for time series forecasting holds great potential, as they can effectively handle uncertainty and capture auto-regressive patterns.

5 Conclusion

In conclusion, our exploration of Mixture Density Networks (MDNs) has unveiled a promising landscape for probabilistic predictions across diverse datasets. By reimagining Bishop's foundational work from 1994, we've demonstrated the adaptability and versatility of MDNs. From predicting inverse sinusoidal patterns in toy datasets to modeling intricate robot kinematics and tackling digit prediction in MNIST, MDNs consistently showcase their ability to capture complex relationships and handle uncertainty. While the road ahead may involve further optimization and fine-tuning, our experiments underscore the practical utility of MDNs in real-world scenarios. As we navigate the nuanced terrain of probabilistic predictions, MDNs emerge as valuable tools with the potential to enhance predictive modeling in a range of applications, laying the groundwork for future advancements in the field.

Future works could provide enhancement on the current implementation that we have done by comparing MDNs with Deep Mixture of Experts [2] [6] for instance.

6 Contribution statement

Antoine Debouchage: Experimentations on the robot kinematics and glimpse of audio data processing using MDN + illustrations and enhancement of figures, report writing.
Valentin Denée: theoretical foundations, beginning of an additional test on time series (additional test not yet finished), report writing.
Clement Wang: Initial implementation of MDN framework, training and visualization on the toy dataset and image regression task, report writing.

References

[1] Christopher M. Bishop. Mixture density networks. Workingpaper, Aston University, 1994.

[2] Shlomo E. Chazan, Jacob Goldberger, and Sharon Gannot. Speech enhancement using a deep mixture of experts, 2017.

[3] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Lara Orlandic, Tomas Teijeiro, and David Atienza. The coughvid crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms. *Scientific Data*, 8(1):156, Jun 2021.

[6] Heiga Zen and Andrew Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3844–3848, 2014.

A Appendix

A.1 Image regression on MNIST dataset

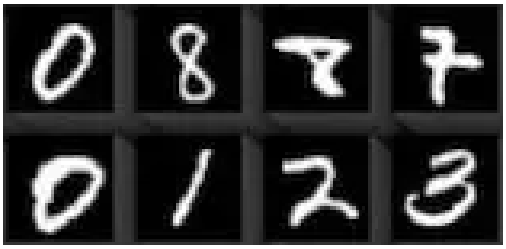


Figure 8: MNIST dataset samples visualization

A.2 Robot kinematics

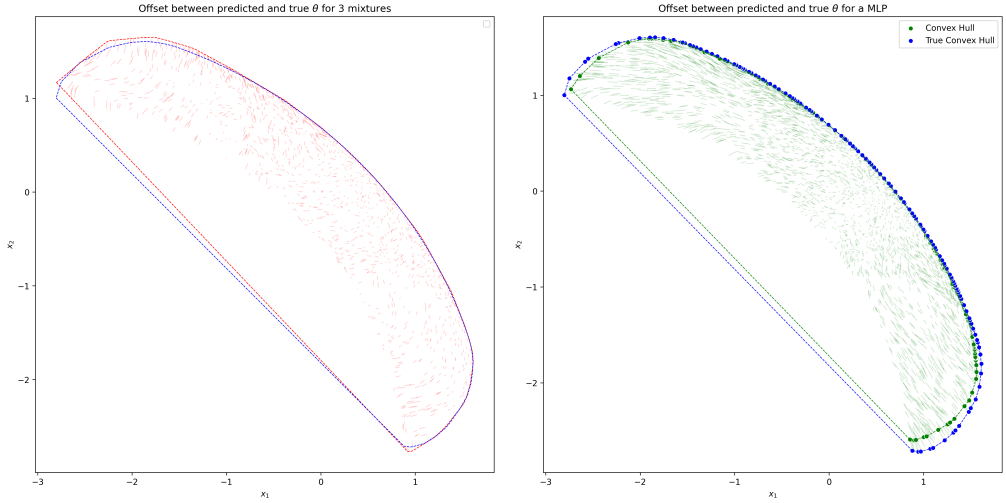


Figure 9: Positioning error as straight lines from desired positions to predicted ones with convex hulls

A.3 Discussion

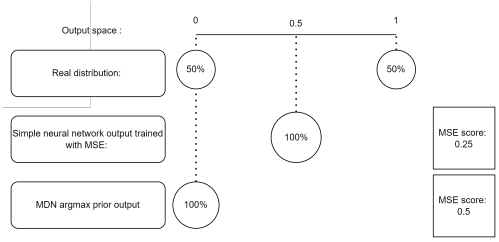


Figure 10: Simple example in the perfect convergence case

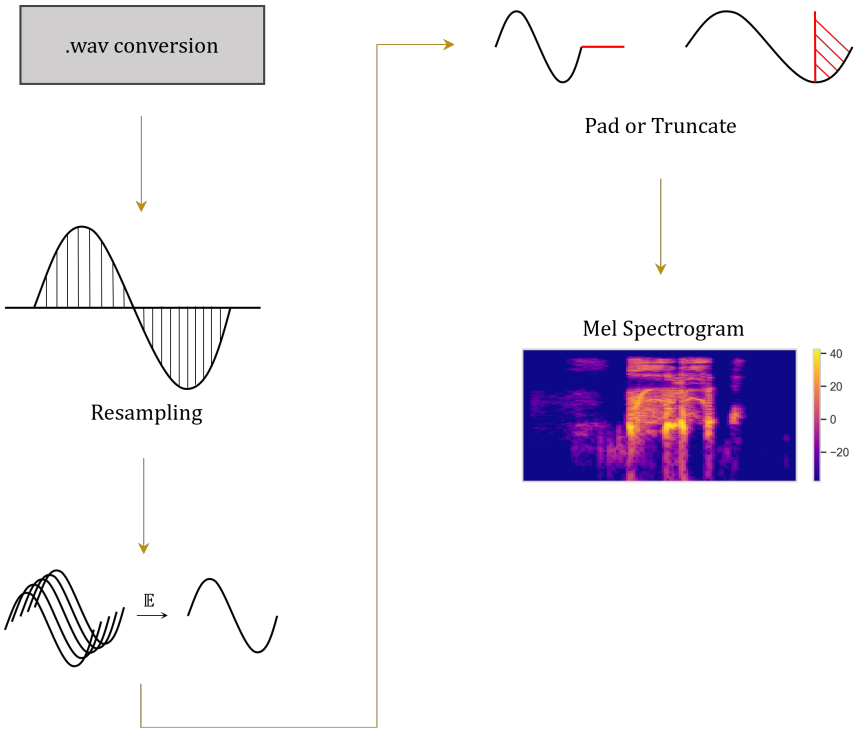


Figure 11: Audio signal preprocessing

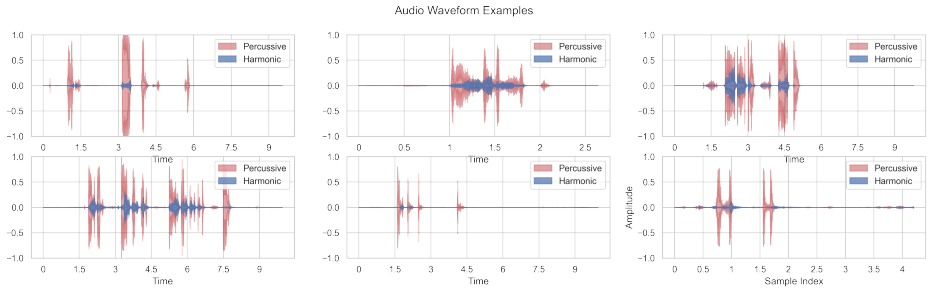


Figure 12: Samples of audio waveform with percussive and harmonic components

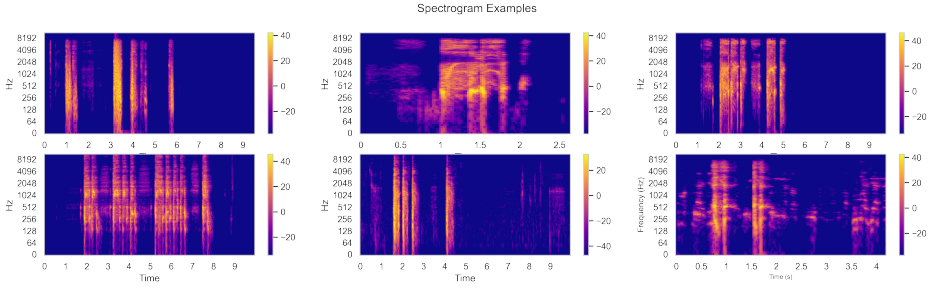


Figure 13: Mel Spectrogram of the samples illustrated in figure 12