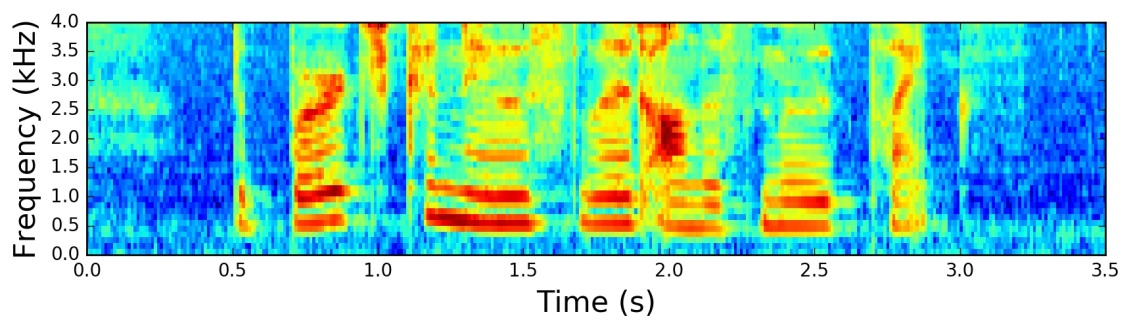
**end to end speech to text model**

1. **sumary:**
2. **Prepare the data pipeline:**
3. **Convert audio waves to Spectrogram :**

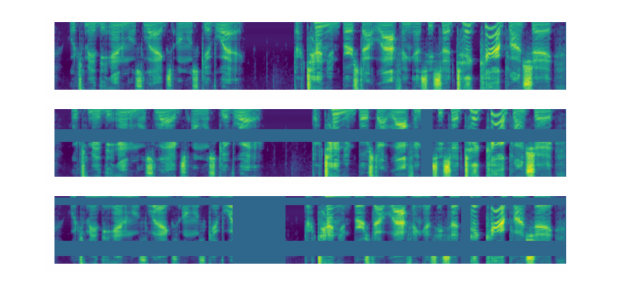
We'll take raw audio waves and transform them into Mel Spectrograms.

you can just think of a Mel Spectrogram as essentially a picture of sound.



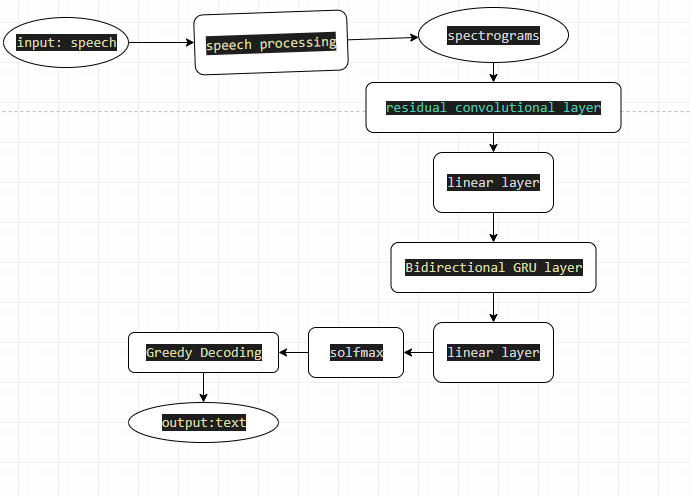
1. **Data augmentation - SpecAugment :**

We found Spectrogram Augmentation (SpecAugment), to be a much simpler and more effective approach. SpecAugment, was first introduced in the paper [SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition](https://arxiv.org/abs/1904.08779?undefined), in which the authors found that simply cutting out random blocks of consecutive time and frequency dimensions improved the models generalization abilities significantly.



**II) model design :**

1. **Diagram :**

****

1. **Detail :**

this model will be similar to the Deep Speech 2 architecture. The model will have two main neural network modules - N layers of Residual Convolutional Neural Networks (ResCNN) to learn the relevant audio features, and a set of Bidirectional Gated Recurrent Units (BiGRU) to leverage the learned ResCNN audio features.

1. **Architecture:**

**+** *ResCNN* :

Convolutional Neural Networks (CNN) are great at extracting abstract features, and we'll apply the same feature extraction power to audio spectrograms. Instead of just vanilla CNN layers, we choose to use Residual CNN layers. Residual connections (AKA skip connections) were first introduced in the paper [Deep Residual Learning for Image Recognition](https://arxiv.org/abs/1512.03385?undefined), where the author found that you can build really deep networks with good accuracy gains if you add these connections to your CNN's. Adding these Residual connections also helps the model learn faster and generalize better. The paper [Visualizing the Loss Landscape of Neural Nets](https://arxiv.org/abs/1712.09913?undefined) shows that networks with residual connections have a “flatter” loss surface, making it easier for models to navigate the loss landscape and find a lower and more generalizable minima.

**+** *BiGRU:*

Recurrent Neural Networks (RNN) are naturally great at sequence modeling problems. RNN's processes the audio features step by step, making a prediction for each frame while using context from previous frames. We use BiRNN's because we want the context of not only the frame before each step, but the frames after it as well. This can help the model make better predictions, as each frame in the audio will have more information before making a prediction. We use Gated Recurrent Unit (GRU's) variant of RNN's as it needs less computational resources than LSTM's, and works just as well in some cases.

+ The model outputs a probability matrix for characters which we'll use to feed into our decoder to extract what the model believes are the highest probability characters that were spoken.

1. **Optimizer and schedual:**

+ *optimizer* : **AdamW**

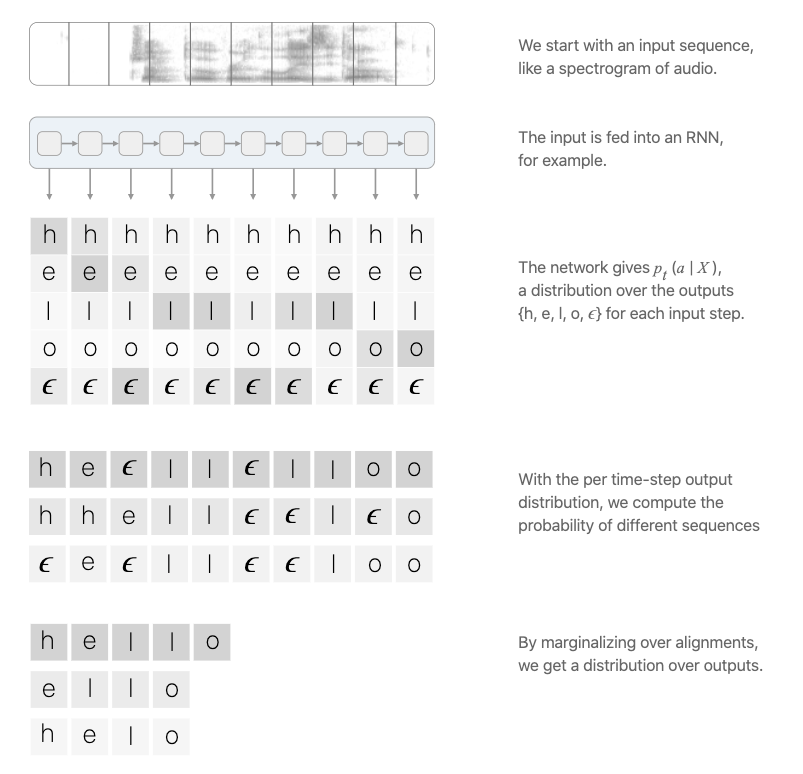
**Adam** is a widely used optimizer that helps your model converge more quickly, therefore, saving compute time, but has been notorious for not generalizing as well as **Stochastic Gradient Descent** AKA **SGD**. **AdamW** was first introduced in [Decoupled Weight Decay Regularization](https://arxiv.org/abs/1711.05101?undefined), and is considered a “fix” to **Adam**.

+ *learning rate schedual* : **One Cycle Learning Rate Scheduler**

1. **Loss function :**

Our model will be trained to predict the probability distribution of all characters in the alphabet for each frame (ie, timestep) in the spectrogram we feed into the model.

+ Traditional speech recognition models would require you to align the transcript text to the audio before training, and the model would be trained to predict specific labels at specific frames .however , the CTC loss function is that it allows us to skip this step. Our model will learn to align the transcript itself during training. The key to this is the “blank” label introduced by CTC, which gives the model the ability to say that a certain audio frame did not produce a character.



1. **Evaluating Model :**

The industry standard is using the Word Error Rate (WER) as the metric. The Word Error Rate does exactly what it says - it takes the transcription your model outputs, and the true transcription, and measures the error between them.

Another useful metric is called the Character Error Rate (CER). The CER measures the error of the characters between the model's output and the true labels. These metrics are helpful to measure how well model performs.

1. **Decoding :**

Model use a "greedy" decoding method to process our model's output into characters that can be combined to create the transcript. A "greedy" decoder takes in the model output, which is a softmax probability matrix of characters, and for each time step (spectrogram frame), it chooses the label with the highest probability. If the label is a blank label, we remove it from the final transcript.

1. **Monitoring :**

[Comet.ml](https://www.comet.ml/site/?undefined) provides a platform that allows deep learning researchers to track, compare, explain, and optimize their experiments and models. Comet.ml has improved our productivity at AssemblyAI and we highly recommend using this platform for teams doing any sort of data science experiments. Comet.ml is super easy to set up. And works with just a few lines of code.