HW3P2 Bootcamp

Utterance to Phoneme Mapping using Sequence Models

Spring 2022

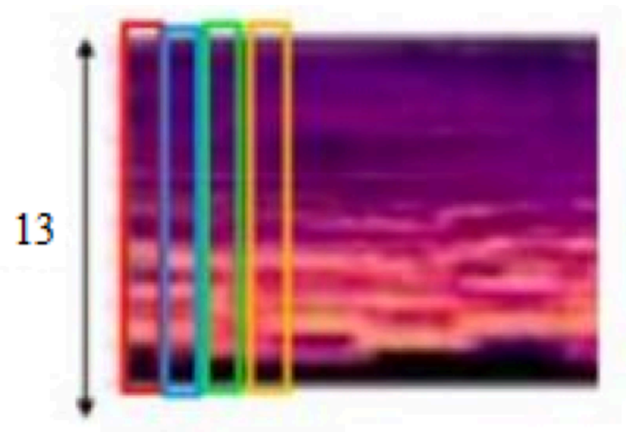
Aparajith Srinivasan

Thanks to Urvil Kenia for helping with the slides and ablation.

Logistics

- Early submission is due **Saturday March 26th, 11:59PM ET**
  - Kaggle submission a with Lev. Dist <= 30
  - Canvas MCQ
- On time submission deadline: **April 7th, 11:59PM ET**
- This part may not take time as much as HW2P2 for training but the high cut-off will be significantly harder
- Constrains:
  - No attention

Problem at hand



Input Utterance MFCC

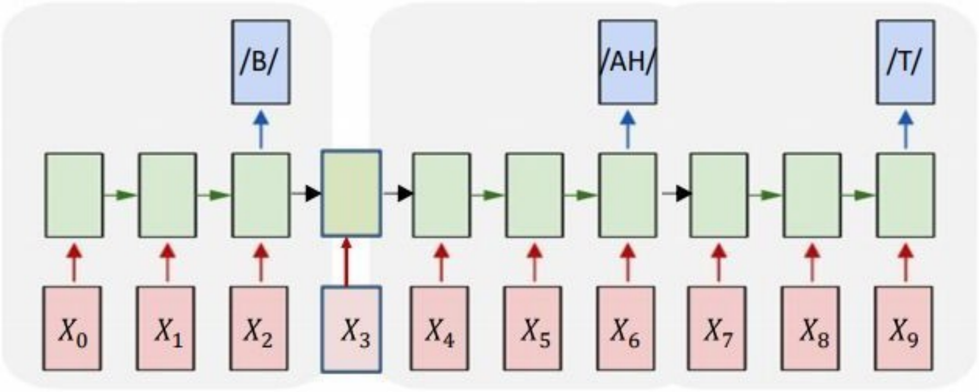
MODEL

[’B’, ‘IH’, ‘K’, ’SH’, ‘A’]

Sequence of Phonemes

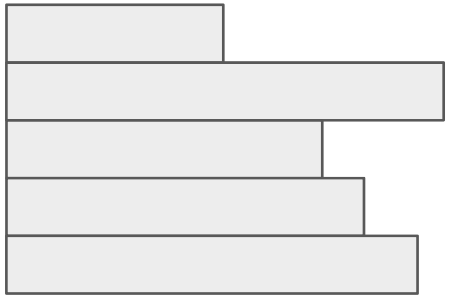
Data and Task

- Features: Same as HW1P2 (13D)
- Labels: Order synchronous but not time synchronous
- Should output sequence of phonemes
  - [‘B’, ‘IH’, ‘K’, ‘SH’, ‘A’] (precisely the indexes)
- Loss: CTCLoss
- Metric: mean Levenshtein distance
  - Can import (given in starter notebook)
  - Sequence of Phonemes -> String and then calculate distance (Use PHONEMES and PHONEMES_MAP)



Batch of Variable Length Inputs: Padding

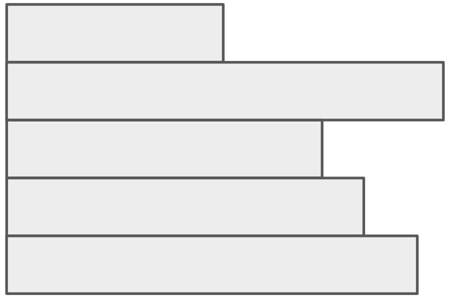
- HW1, HW2: Equal length inputs
- HW3: Variable Length sequences
- Steps:
  - Padding
  - Packing



Ref: 11785 Fall 21 Bootcamp

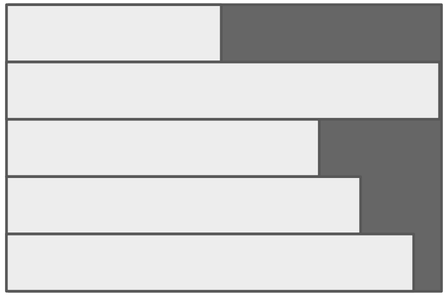
Batch of Variable Length Inputs: Padding

- Padding



Need to store unpadded lengths as well.
Have the variables *lengths_x, lengths_y* in the starter notebook

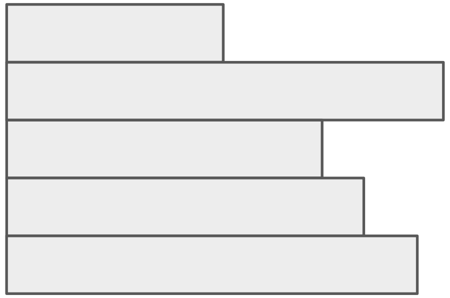
Padded to equal lengths



Ref: 11785 Fall 21 Bootcamp

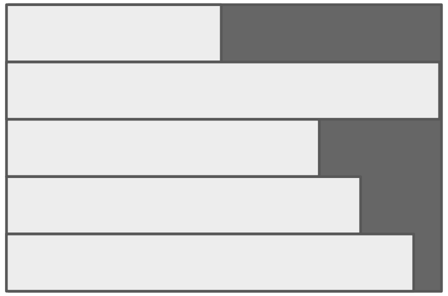
Batch of Variable Length Inputs: Padding

- Padding



Need to store unpadded lengths as well.
Have the variables *lengths_x, lengths_y* in the starter notebook

Padded to equal lengths

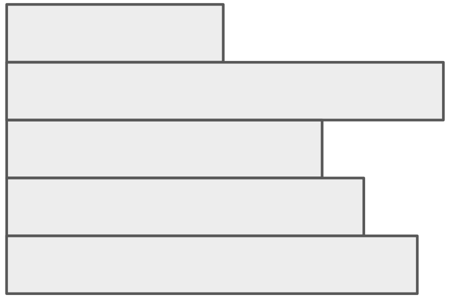


(B, *, 13) → (B, T, 13)

Ref: 11785 Fall 21 Bootcamp

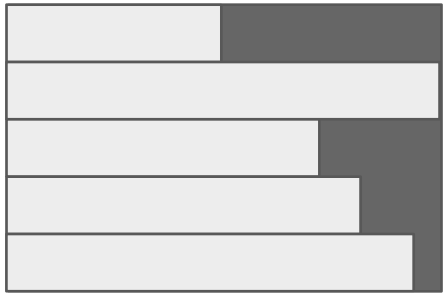
Batch of Variable Length Inputs: Padding

- Padding



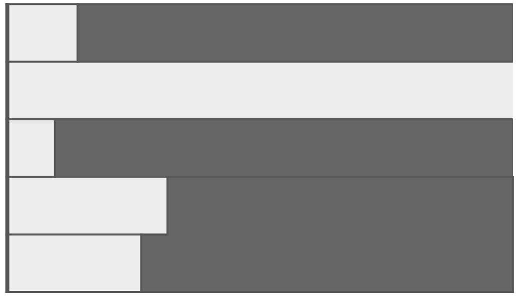
Need to store unpadded lengths as well.
Have the variables *lengths_x, lengths_y* in the starter notebook

Padded to equal lengths



(B, *, 13) → (B, T, 13)

- Not for the whole dataset (instead we pack after padding)



Ref: 11785 Fall 21 Bootcamp

Batch of Variable Length Inputs: Packing

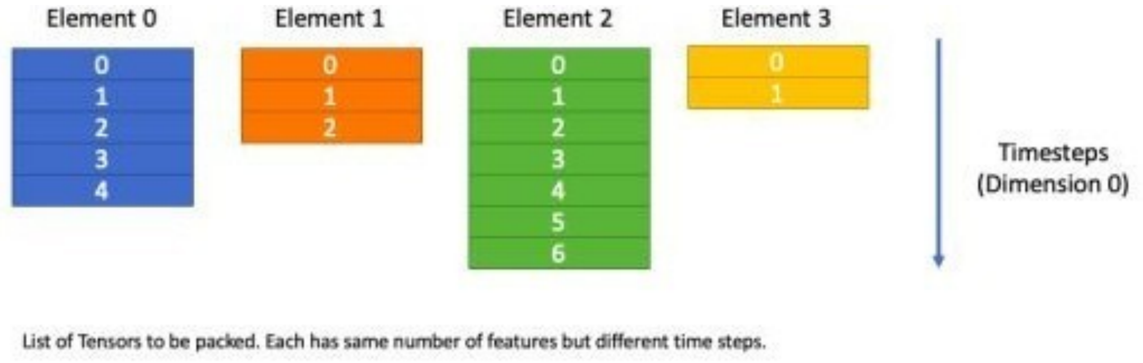


Figure 2: List of tensors we want to pack

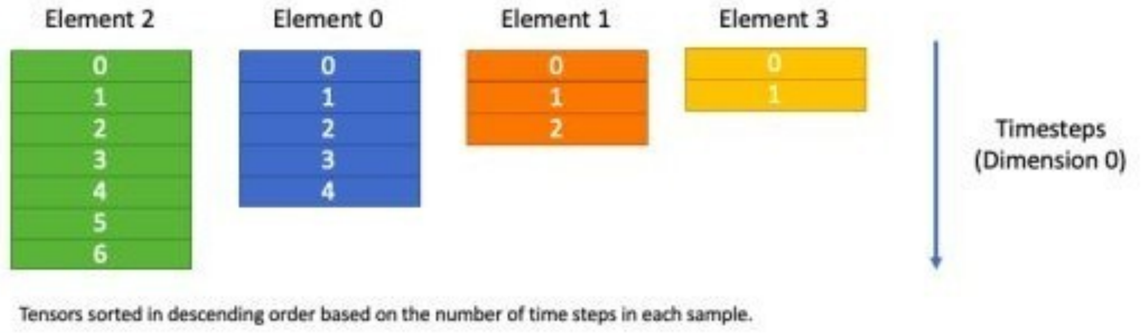


Figure 3: First we sort the list in a descending order based on number of timesteps in each

Ref: 11785 Fall 21 Bootcamp

Batch of Variable Length Inputs: Packing

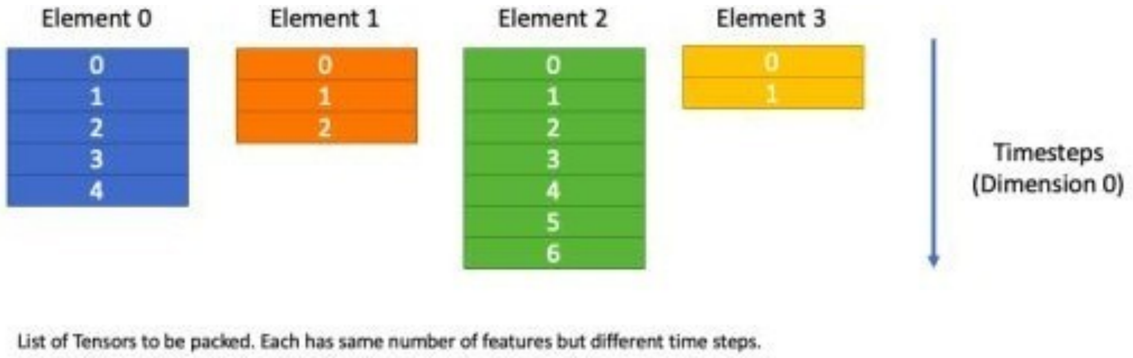


Figure 2: List of tensors we want to pack

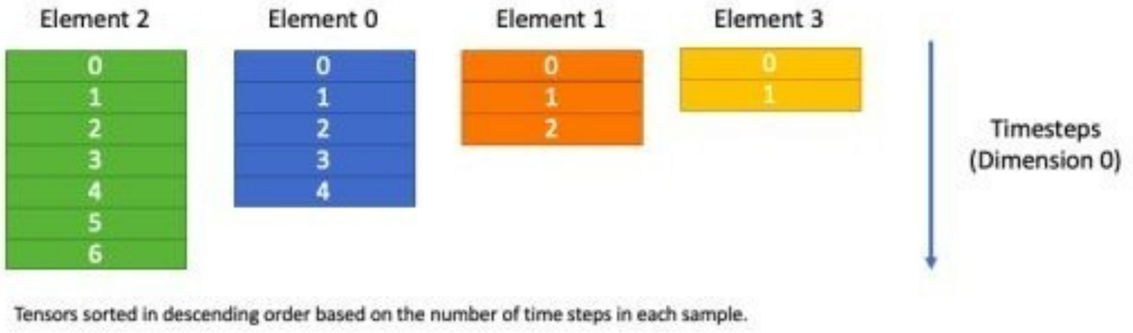


Figure 3: First we sort the list in a descending order based on number of timesteps in each

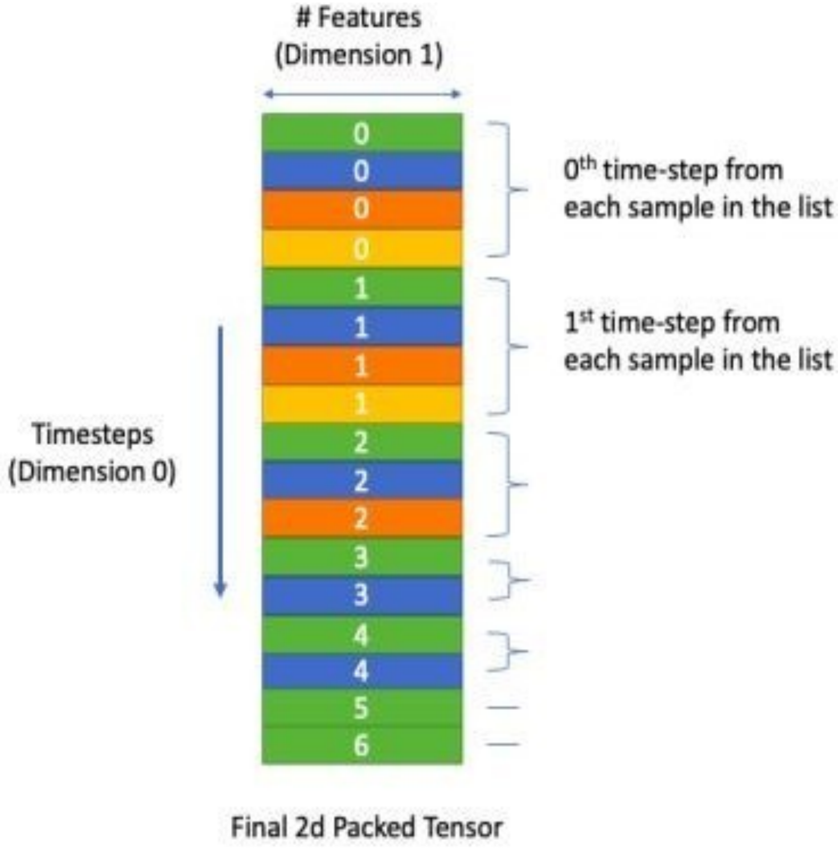
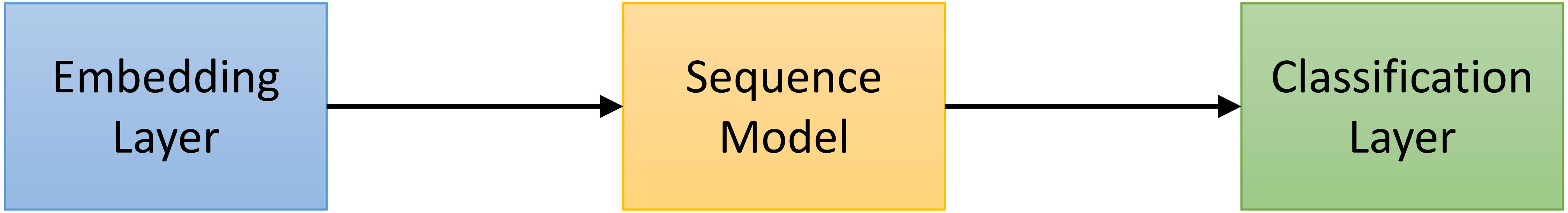


Figure 4: Final Packed 2d Tensor

Ref: 11785 Fall 21 Bootcamp

Parts of a Sequence Model

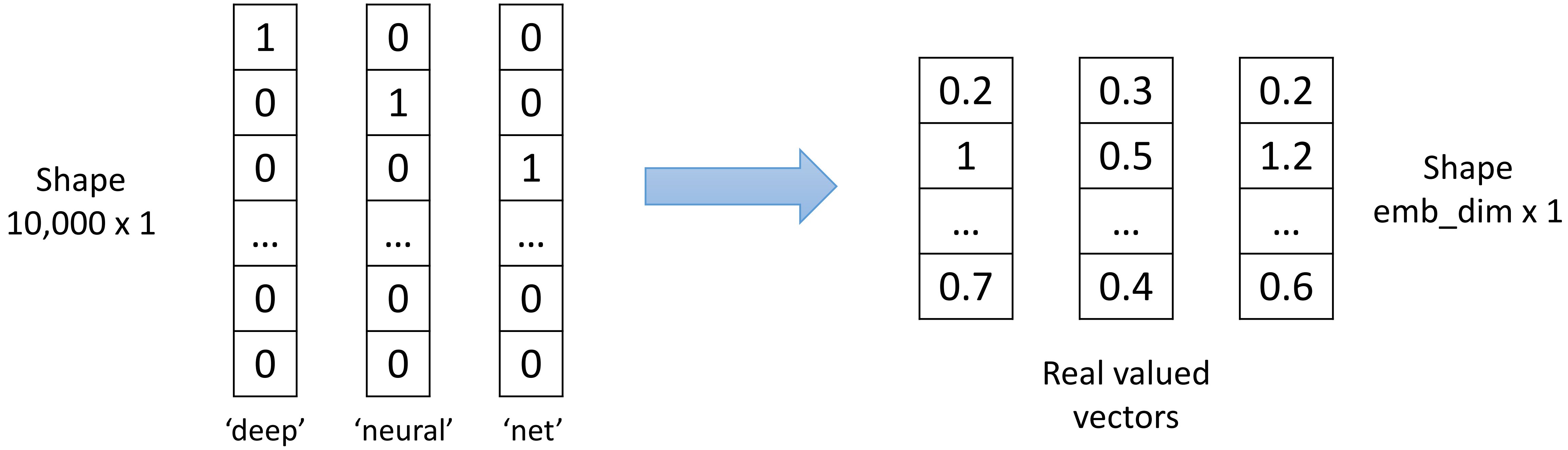


Embedding Layer

- Optional but recommended
- Used to increase/decrease the dimensionality of the input

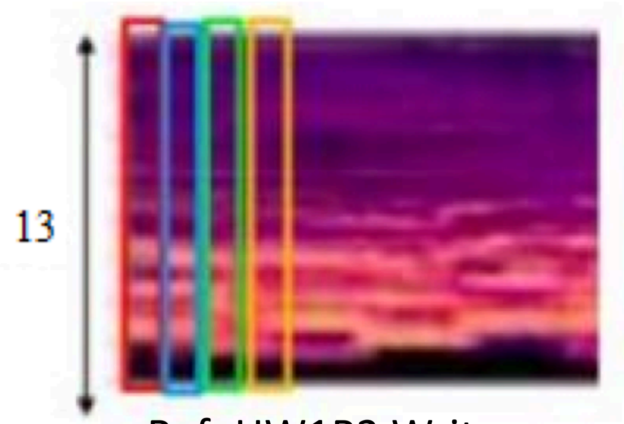
Embedding Layer

- Optional but recommended
- Used to increase/decrease the dimensionality of the input
- Eg. In NLP, 10k vocabulary represented as 1 hot vectors with 10k dim



Embedding Layer

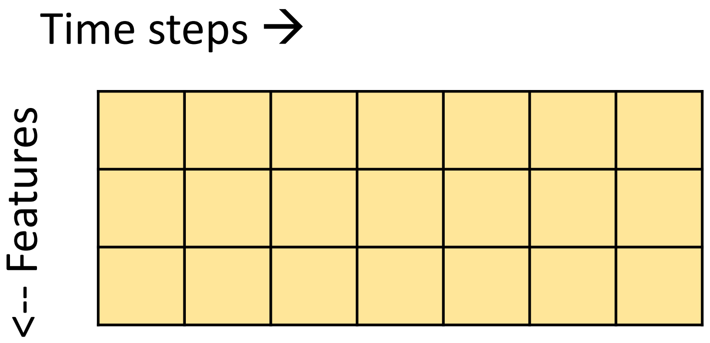
- Optional but recommended
- Used to increase/decrease the dimensionality of the input
- Our task:
  - Input dim = 13
  - Expand to emb_dim > 13 for feature extraction



Ref: HW1P2 Write-up

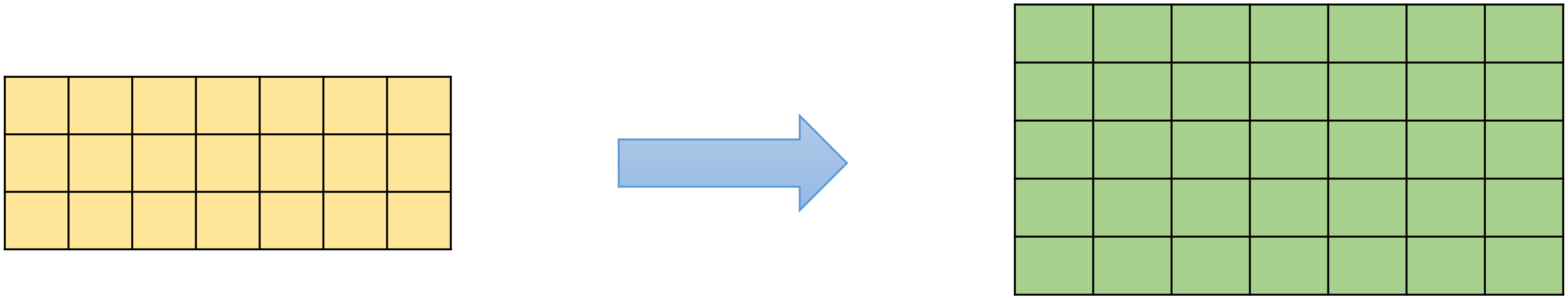
Embedding Layer: Conv1d Layers

- Consider the below as an input having 3 features at each time instant



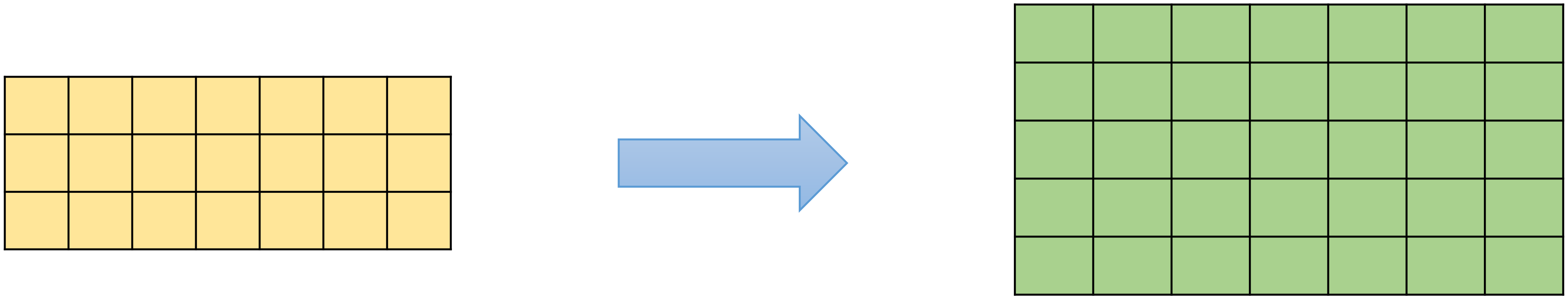
Embedding Layer: Conv1d Layers

- We can use Convolution to which increases the channels of the input as we go deeper.



Embedding Layer: Conv1d Layers

- We can use Convolution to which increases the channels of the input as we go deeper.

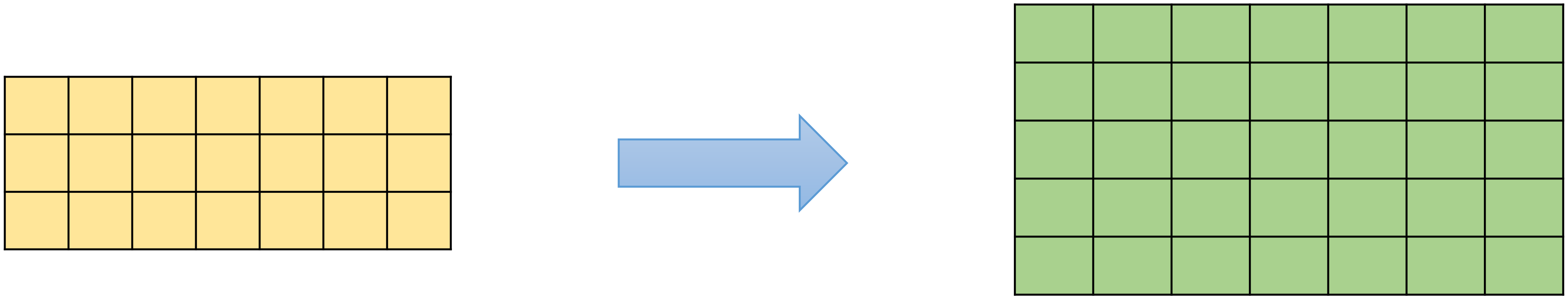


- No. Filters = 5
- Kernel= 3; Padding= 1; Stride= 1
- Kernel= 5; Padding= 2; Stride= 1

(Or anything similar)

Embedding Layer: Conv1d Layers

- We can use Convolution to which increases the channels of the input as we go deeper.



- No. Filters = 5
- Kernel= 3; Padding= 1; Stride= 1
- Kernel= 5; Padding= 2; Stride= 1

(Or anything similar)

3D → 5D

Embedding Layer: Conv1d Layers

- Our input is of shape (B, T, 13) (after padding). How can we change it to (B, T, 64) ?

Assuming *batch_first = True* (You may also have it as (T, B, 13)

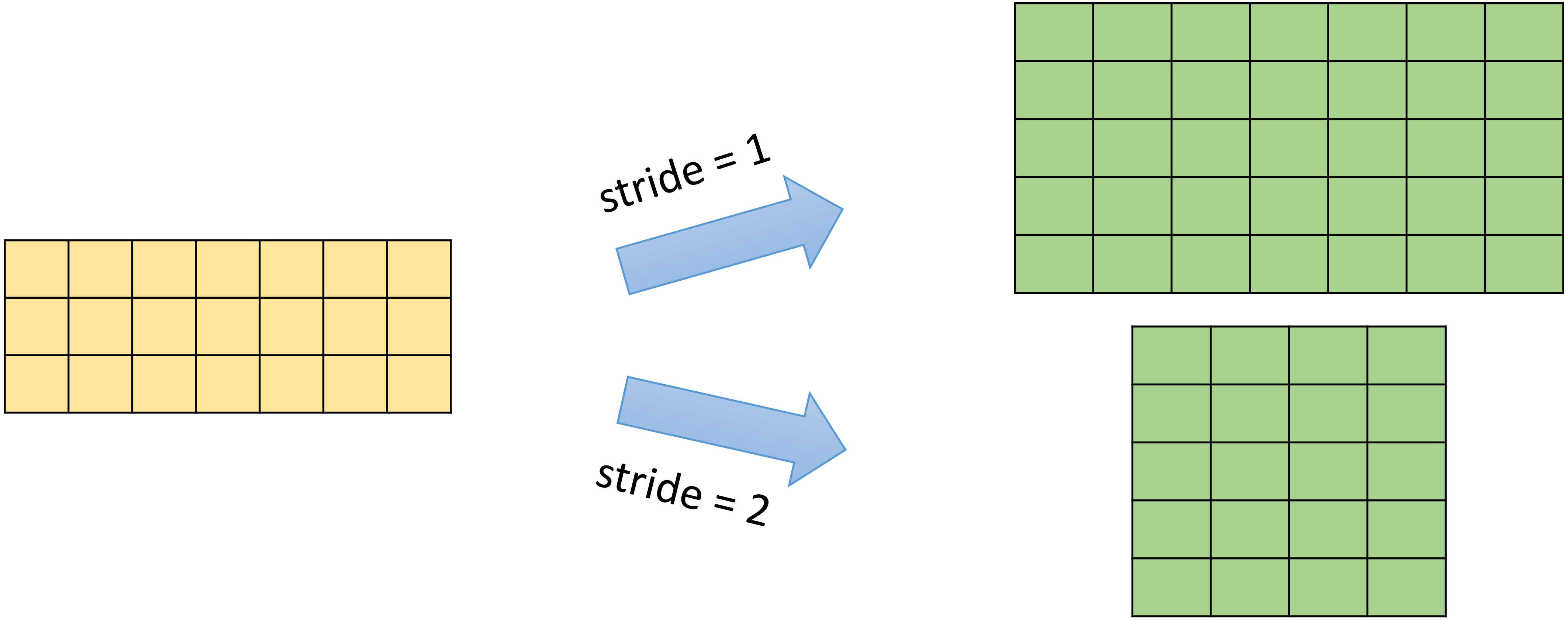
Embedding Layer: Conv1d Layers

- Our input is of shape (B, T, 13) (after padding). How can we change it to (B, T, 64) ?
- Transpose/Permute: (B, T, 13) → (B, 13, T) which makes #channels = 13 (Conv1d)
- Apply convolution (B, 13, T) → (B, 64, T)
- Transpose/Permute: (B, 64, T) → (B, T, 64) (pack and pass to LSTM/GRU)
- Note: This is done in the forward function

Assuming *batch_first = True* (You may also have it as (T, B, 13)

Embedding Layer: Conv1d Layers

If stride > 1, we effectively reduce the time steps



Embedding Layer: Conv1d Layers

- Stride > 1 reduces computation for LSTM and training is faster.
- However, too much reduction in time steps will lead to loss of information (we don’t recommend downsampling more than 4x)

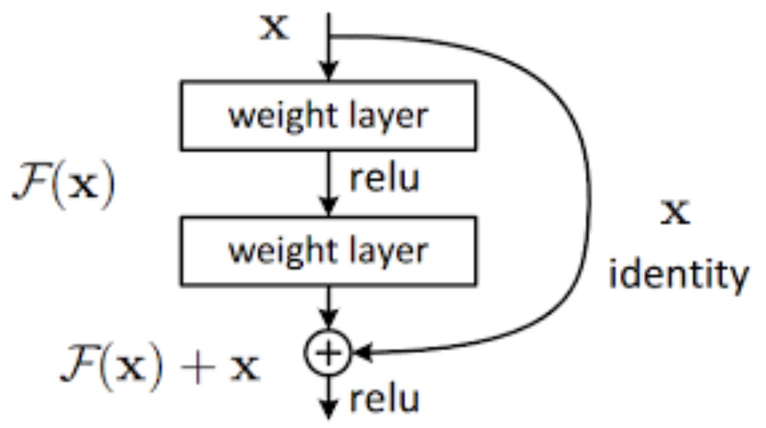
Embedding Layer: Conv1d Layers

- Stride > 1 reduces computation for LSTM and training is faster.
- However, too much reduction in time steps will lead to loss of information (we don’t recommend downsampling more than 4x)

- **Note: Stride > 1 alters number of time steps. You need to change lengths_x accordingly**
  - Use convolution formula ***(X – K + 2*P)//S*** (or)
  - Clamp lengths to length of embedding (torch function)

Embedding Layer: Conv1d Layers

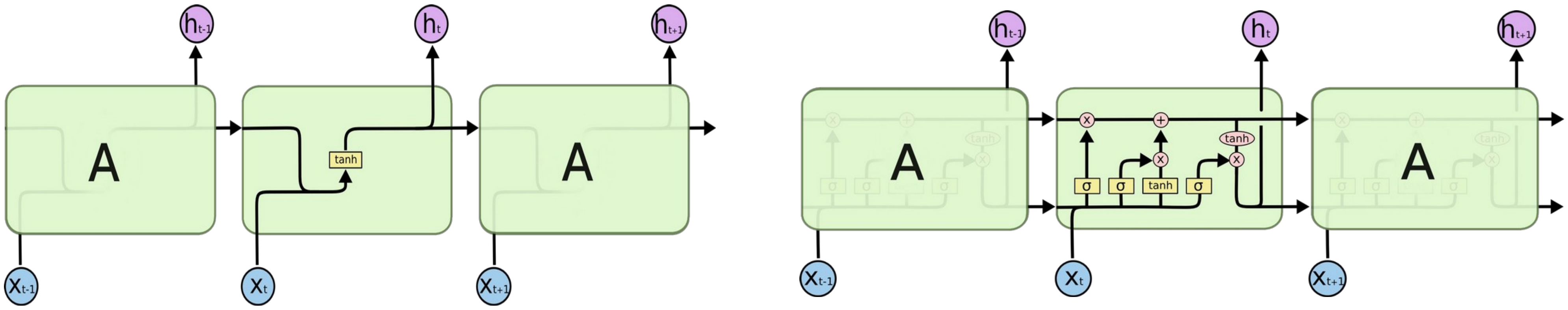
- You can try convolution layers based on residual blocks
- Our observation: Deeper embedding layers without skip connections are not so fruitful
- Hint: Remember HW2P2!



https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Sequence Model

- Can use RNN, GRU, LSTM (recommended) from *torch.nn*



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Sequence Model

- Important parameters/hyper parameters in *nn.LSTM()*
  - *input_size (13 or emb_size)*
  - *hidden_dim*
  - *num_layers*
  - *dropout*
  - *bidirectional*
  - Note: when *bidirection = True*, LSTM outputs a shape of *hidden_dim* in the forward direction and *hidden_dim* in the backward direction
    (in total, *2*hidden_dim*)

Classification Layer

- Same as HW1P2
- Output from the sequence model goes to the classification layer
- Variations
  - Deeper
  - Wider
  - Different activations
  - Dropout

Hyperparameters and Regularization

- In this HW,

ARCHITECTURES >> HYPERPARAMETERS

- Don't stick with one architecture and vary the hyperparameters

***** The following suggestions might or might not work. You may want to run a proper ablation study as suggested in the previous homeworks*****

Hyperparameters and Regularization

- Cepstral Normalization:

$$X \rightarrow (X - \text{mean})/\text{std}$$

- Different weight initialization (for Conv and Linear layers)
- Weight decay with optimizer

Hyperparameters and Regularization

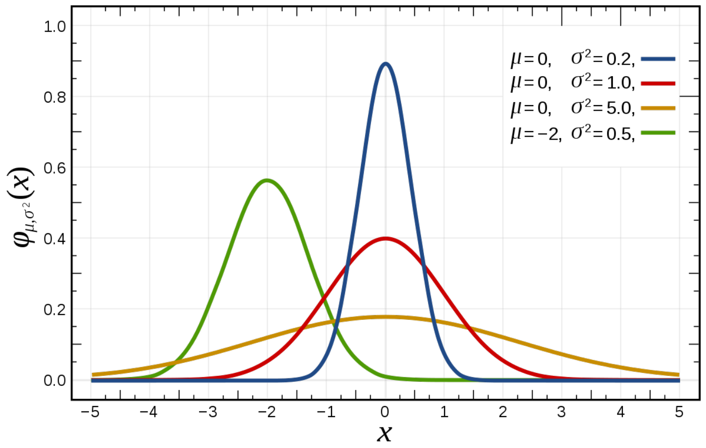
- Scheduler is very important
  - ReduceLRonPlateau (Most of our ablation)
    - Lev distance might start to oscillate at lower values
    - Can have a somewhat higher patience
  - Cosine Annealing
    - Try with higher number of epochs

Hyperparameters and Regularization

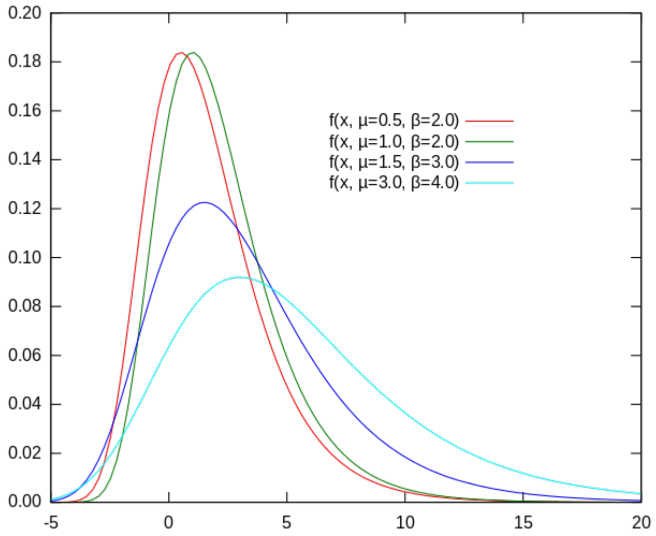
- Dropout is key
  - Can use dropout in all the 3 layers: Embedding, Sequence model and classification
  - You can also start with a small dropout rate and increase after the model gets trained
- Locked Dropout for LSTM layer

Hyperparameters and Regularization

- Addition of Noise *(only during training)*
  - Gaussian Noise
  - Gumbel Noise
- Need not add to all samples. Implement your module *AddNoise(nn.module)* in such a way that it adds noise to random inputs



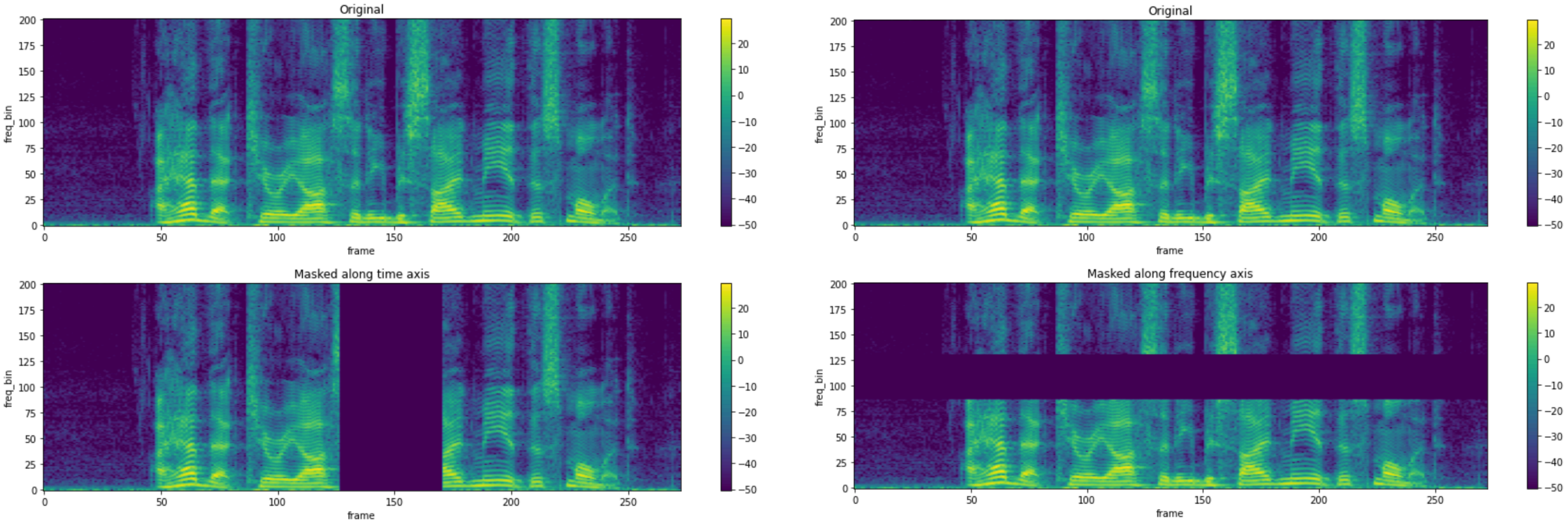
https://en.wikipedia.org/wiki/Normal_distribution



https://en.wikipedia.org/wiki/Gumbel_distribution

Hyperparameters and Regularization

- Torch Audio Transforms [docs]
  - Time Masking
  - Frequency Masking



Hyperparameters and Regularization

- Beam width
  - Higher beam width may give better results (advisable to keep test beam width below 50 for computation purposes)
  - Sometimes bw = 1 (greedy search) also gives good results
  - Tip: Don’t use a high beam width while validating in each epoch (time per epoch will be higher)

Final Tips

- More work by varying architectures
- Make proper ablation by varying just one parameter/hyperparameter to observe its influence
- Have multiple notebooks running:
  - Colab Pro users: 1 with high ram and 3 with standard ram
  - AWS: Can run multiple notebooks when some GPU memory is left
- Private leader board is worse (gives at least 0.1 higher distance than public)

Final Tips

- Make sure to split work within your study groups

Group 52 ablation

	Name	Dataset	Context	Layers	BatchNorm	Dropout rate	Activation	lr
Example	Aparajith	Full/25%	16	[429, 512, 512, 256, 40]	All layers	0.25	ReLU	0.001
1	John M	Full	28	[741, 2048, 1024, 512, 256, 40]	All layers	0.1	ReLU	0.001
2	Bobby	Full	24	[637, 1024, 512, 256, 40]	Every other layer	0.1	ReLU	0.005
3	Bobby	Full	24	[637, 1024, 512, 256, 40]	Every other layer	0.2	ReLU	0.005
4	Bobby	Full	24	[637, 2048, 1024, 512, 256, 40]	Every other layer	0.1	ReLU	0.005
5	Jon D	Full	16	[429, 1024, 512, 256, 40]	All layers	0.1	ReLU	0.001
6	Bobby	Full	48	[1261, 8192, 4096, 2048, 1024, 512, 256, 40]	All layers	0.2	ReLU	0.005
7	John M	Full	32	[845, 8192, 4096, 2048, 1024, 512, 256, 40]	All layers	0.1	ReLU	0.005
8	Jon D	Full	24	[637, 1024, 512, 256, 40]	All layers	0.1	ReLU	0.001
9A.1	Bobby	8192	24	[637, 4096, 2048, 1024, 512, 40]	All layers	0.3	ReLU	0.001
9A.2	Bobby	8192	24	[637, 4096, 2048, 1024, 512, 40]	All layers	0.3	ReLU	0.001
9A.3	Bobby	8192	24	[637, 4096, 2048, 1024, 512, 40]	All layers	0.3	ReLU	0.001
9A.4	Bobby	Full	24	[637, 4096, 2048, 1024, 512, 40]	All layers	0.3	ReLU	0.001
9B.1	John M	8192	32	[xxx, 4096, 2048, 1024, 512, 40]	All Layers	0.3	ReLU	0.001
9B.2	John M	8192	24	[xxx, 4096, 2048, 1024, 512, 40]	All Layers	0.4	ReLU	0.001
9B.3	John M	8192	24	[xxx, 4096, 2048, 1024, 512, 40]	All Layers	0.4	ReLU	0.001
9B.4	John M	8192	24	[xxx, 4096, 2048, 1024, 512, 40]	All Layers	0.2	ReLU	0.001
9C.1	Jon D	8192	16	[429, 4096, 2048, 1024, 512, 40]	All Layers	0.3	ReLU	0.001
9C.2	Jon D	8192	16	[429, 4096, 2048, 1024, 512, 40]	All Layers	0.3	Leaky ReLU	0.001
9C.3	Jon D	8192	16	[429, 4096, 2048, 1024, 512, 40]	All Layers	0.3	Leaky ReLU	0.001
9D.1	John M	8192	24	[xxx, 4096, 2048, 1024, 256, 40]	All Layers	0.3	ReLU	0.001

- Start Early - High cut-off is tougher than last homework

Medium Cut-off Architecture

Medium Cut-off Architecture

- Embedding: 2 Conv1d Layers (Final emb size 256)
- Sequence model: 4 layer Bi-directional LSTM with dropout (256)
- Classification: 2 Linear layers (2048, 41)
- Optimizer: Adam (lr = 2e-3) with a scheduler
- Epochs: 50 – 100
- Beam width: 30 - 50 (Only for testing)

All the best!