Multimodal Deep Learning

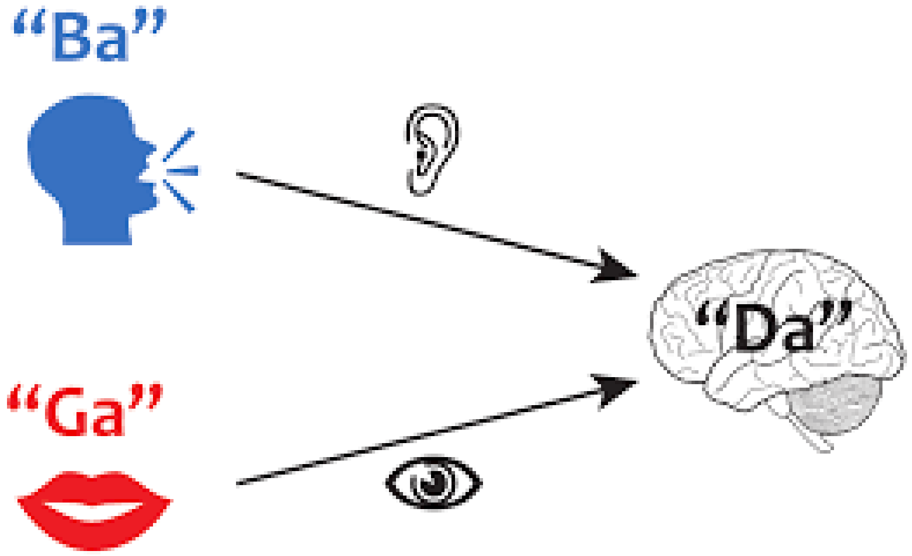
Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, Andrew Y. Ng

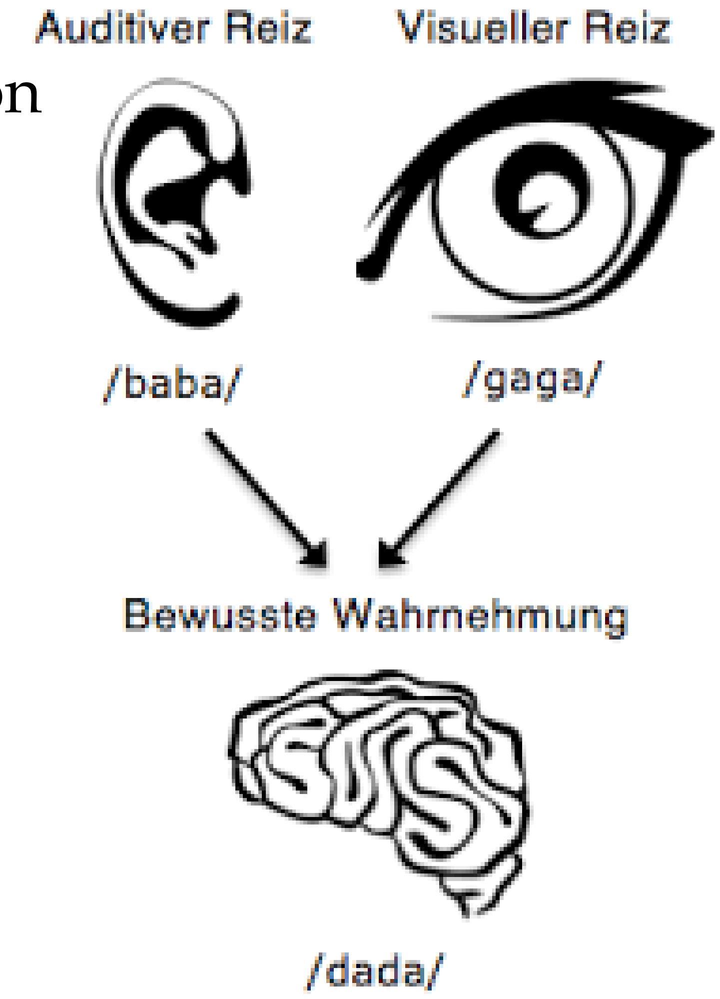
Keywoong Bae, Inha University

I. Introduction

McGurk Effect?

- a perceptual phenomenon that demonstrates an interaction between hearing and vision in speech perception
- a visual */ga/* with a voiced */ba/* is perceived as */da/*



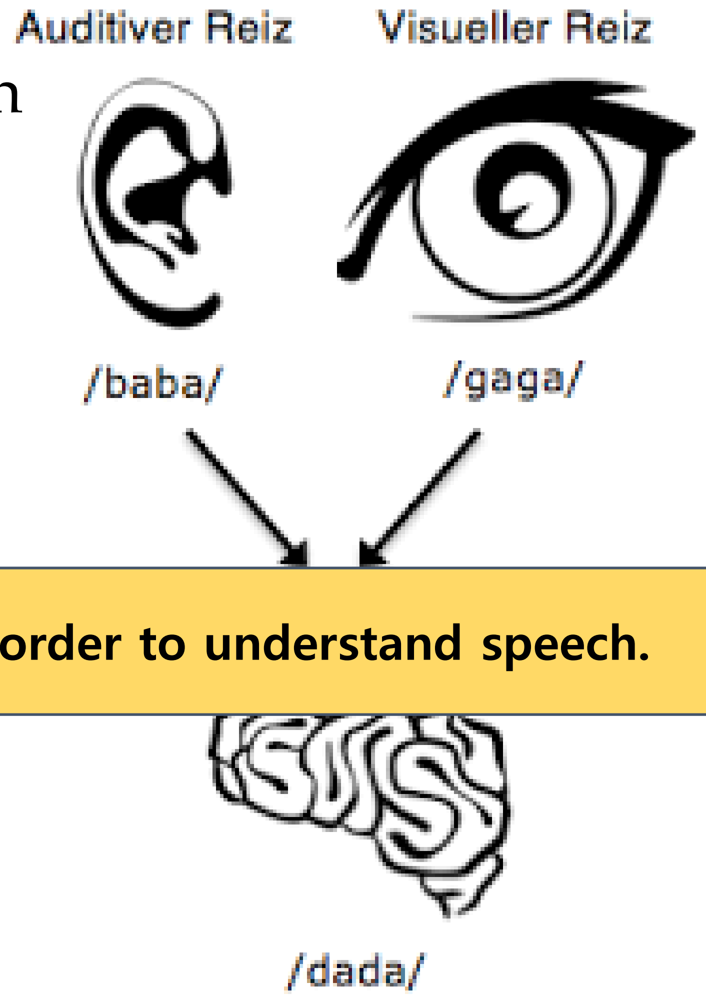


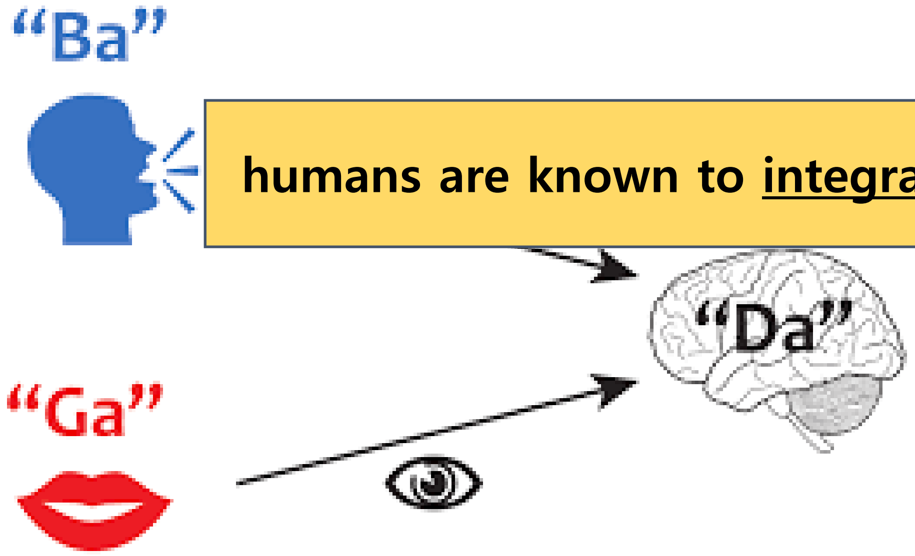
I. Introduction

McGurk Effect?

- a perceptual phenomenon that demonstrates an interaction between hearing and vision in speech perception

- a visual */ga/* with a voiced */ba/* is perceived as */da/*





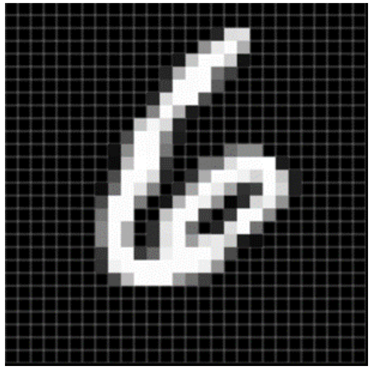
humans are known to <u>integrate</u> audio-visual information in order to understand speech.

I. Introduction

Modeling "mid-level" relationships

- In this paper, they are interested in modeling **<u>"mid-level" relationships</u>**

- "Mid-level" relationship? : **<u>indirectly</u>** related to each other



3-d depth scan

First-order relationship
(depth discontinuities often manifest as strong edges in images)

Image pixels

H E L L O

Phonemes (audio)

"mid-level" relationship
(hard to relate raw pixels to audio waveforms or spectrograms

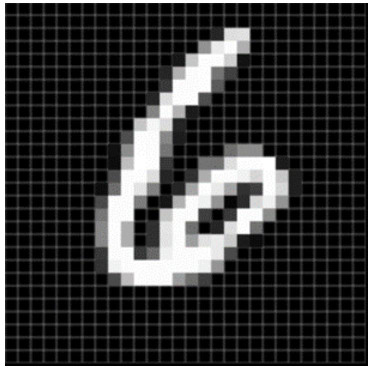
Visemes (video)

I. Introduction

Modeling “mid-level” relationships

- In this paper, they are interested in modeling **<u>“mid-level” relationships</u>**

- “Mid-level” relationship? : **<u>indirectly</u>** related to each other



3-d de[illegible] scan

[illegible]st-order relationship
(depth discontinuities often [illegible]anifest as strong edges in [illegible]ages)

image [illegible]xels

H E L L O

Phonemes (audio)

“mid-level” relationship
(hard to relate raw pixels to audio waveforms or spectrograms

Visemes (video)

I. Introduction

Three tasks and Three learning settings

Three tasks

Three learning settings

	Feature Learning	Supervised Training	Testing
Classic Deep Learning	Audio	Audio	Audio
	Video	Video	Video
Multimodal Fusion	A + V	A + V	A + V
Cross Modality Learning	A + V	Video	Video
	A + V	Audio	Audio
Shared Representation Learning	A + V	Audio	Video
	A + V	Video	Audio

Figure 1: Multimodal Learning settings where A+V refers to Audio and Video.

I. Introduction

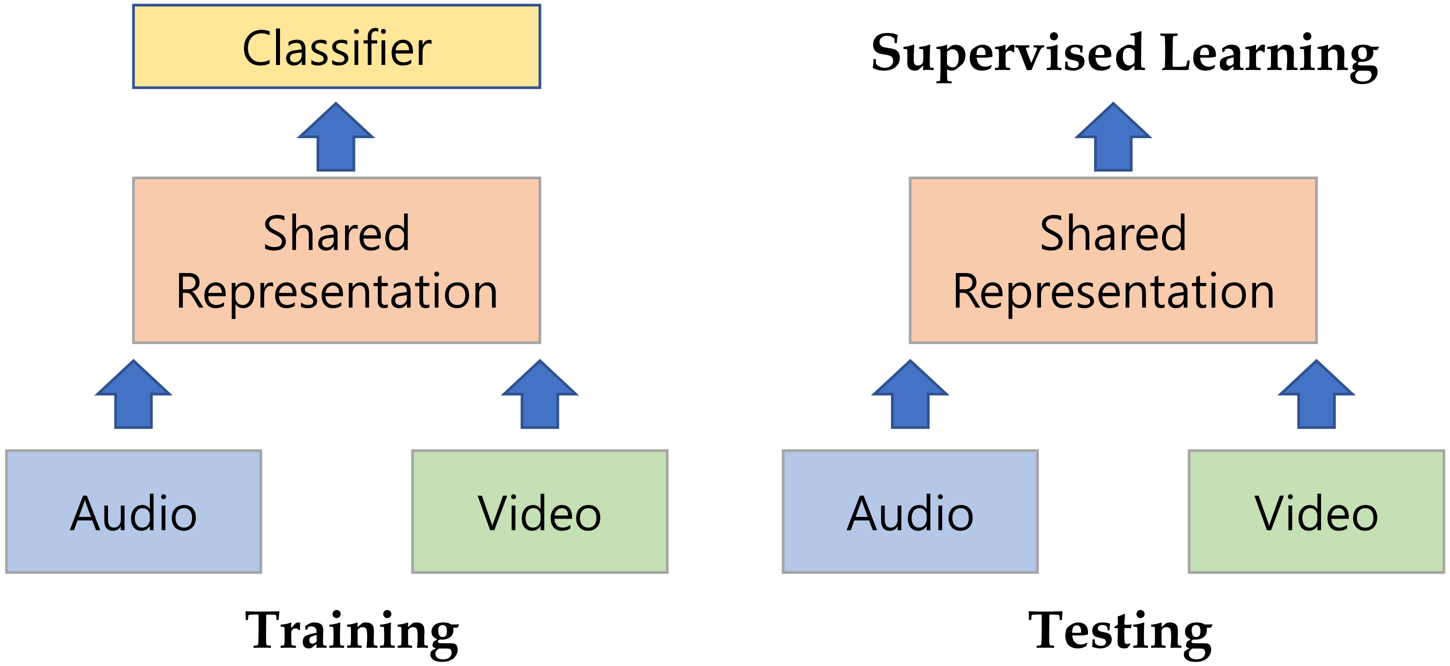
Three learning settings

	Feature Learning	Supervised Training	Testing
Multimodal Fusion	A + V	A + V	A + V

- Multimodal Fusion
- Cross Modality Learning
- Shared Representation Learning

Learning Setting 1) **Multimodal Fusion**

- Use data from all modalities at all tasks
- Typical setting



I. Introduction

Three learning settings

	Feature Learning	Supervised Training	Testing
Cross Modality Learning	A + V	Video	Video
	A + V	Audio	Audio

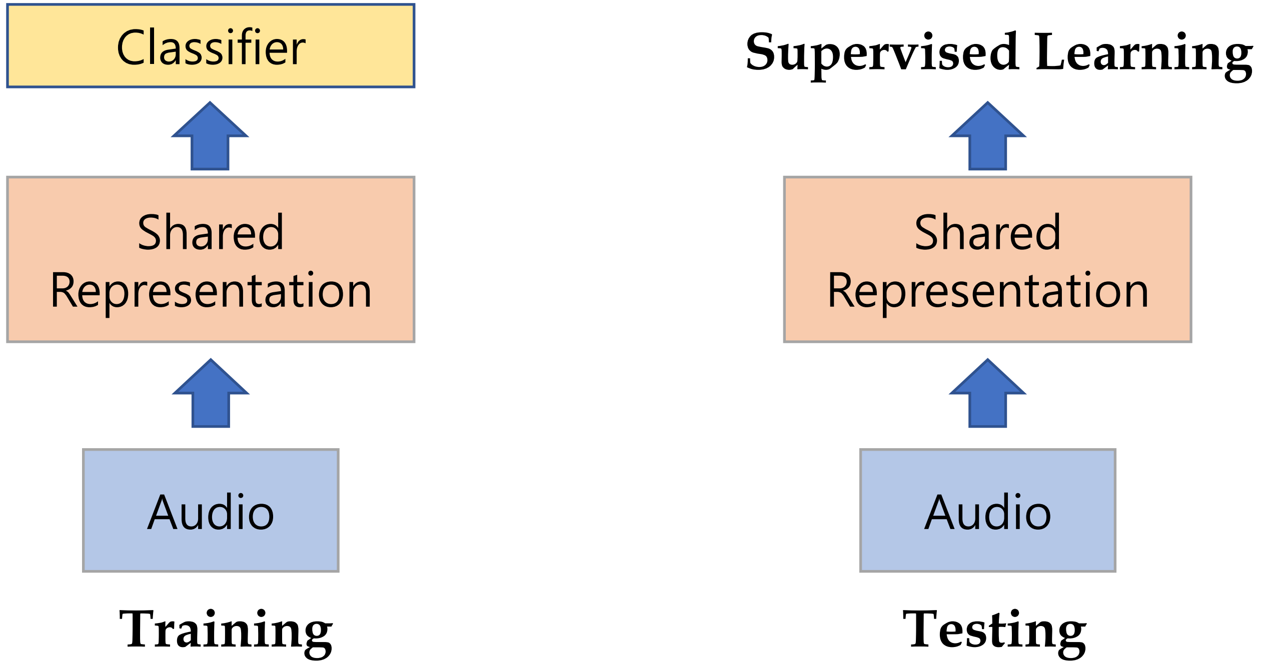
Multimodal Fusion

Cross Modality Learning

Shared Representation Learning

Learning Setting 2) **Cross Modality Learning**

- Data from all modalities is available only feature learning
- Only data from a single modality in training and testing
- Purpose : Learn better single modality representations given unlabeled data from multiple modalities



I. Introduction

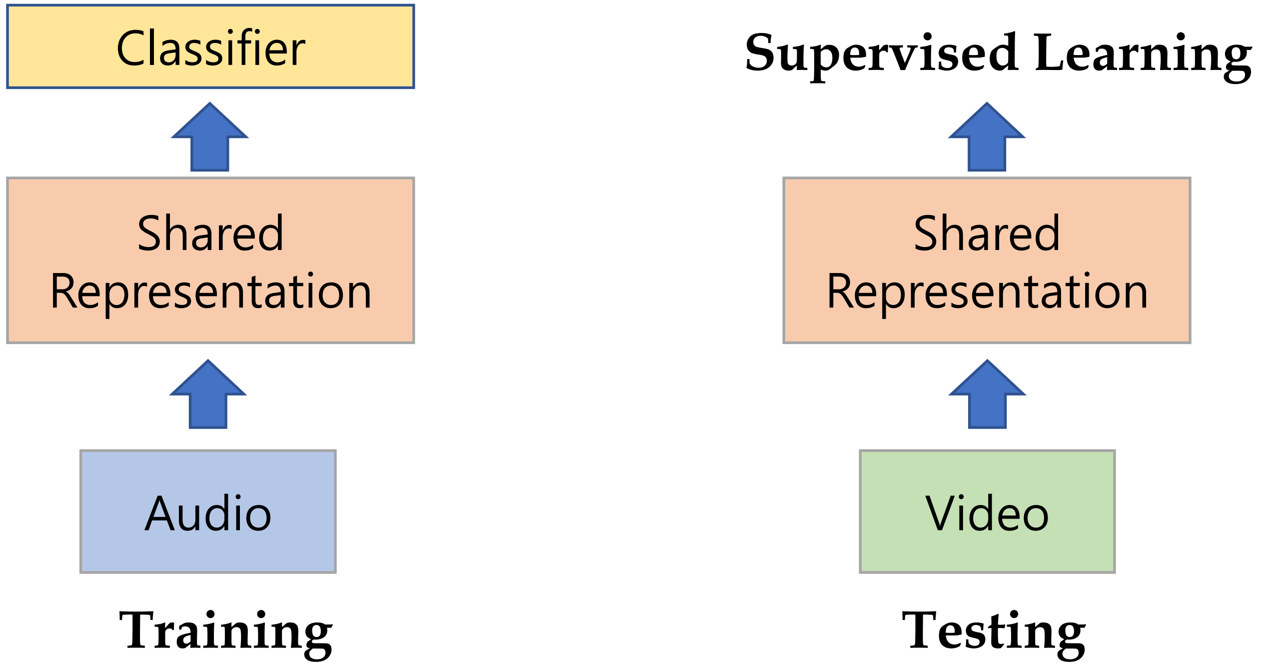
Three learning settings

	Feature Learning	Supervised Training	Testing
Shared Representation Learning	A + V	Audio	Video
	A + V	Video	Audio

- Multimodal Fusion
- Cross Modality Learning
- Shared Representation Learning

Learning Setting 3) **Shared Representation Learning**

- Evaluate if the feature representations can capture correlations across different modalities
- Access whether the learned representations are modality-invariant



II. Background

Sparse Restricted Boltzmann machines

RBM (Restricted Boltzmann machines)

- Simple architecture

- An **undirected graphical model** with hidden variables(h) and visible variables(v)

- **No connections within** hidden variables or visible variables
  (1) express the combination of probability distributions by assuming **independence** between events (conditional probability)
  (2) $p(h, v)$ is complicated to calculate, but $p(v \mid h), p(h \mid v)$is easier to calculate (one value is fixed)

- Unsupervised Learning
  Focus on which features of the input are important and how to combine them to form a pattern.
  (similar to autoencoder)

II. Background

Sparse Restricted Boltzmann machines

RBM (Restricted Boltzmann machines)

- Simple architecture

- An **undirected graphical model** with hidden variables(h) and visible variables(v)

- **<u>No connections within</u>** hidden variables or visible variables
  (1) express the combination of probability distributions by assuming **<u>independence</u>** between events (conditional probability)
  (2) $p(h, v)$ is complicated to calculate, but $p(v \mid h), p(h \mid v)$is easier to calculate (one value is fixed)

$$\boxed{\begin{aligned}&-\log P(\mathbf{v},\mathbf{h}) \propto E(\mathbf{v},\mathbf{h}) = \\ &\qquad \frac{1}{2\sigma^2}\mathbf{v}^T\mathbf{v} - \frac{1}{\sigma^2}\left(\mathbf{c}^T\mathbf{v} + \mathbf{b}^T\mathbf{h} + \mathbf{h}^T W\mathbf{v}\right)\end{aligned}}$$

probability over h, v

$$\boxed{p(h_j|\mathbf{v}) = sigmoid(\frac{1}{\sigma^2}(b_j + \mathbf{w}_j^T\mathbf{v}))}$$

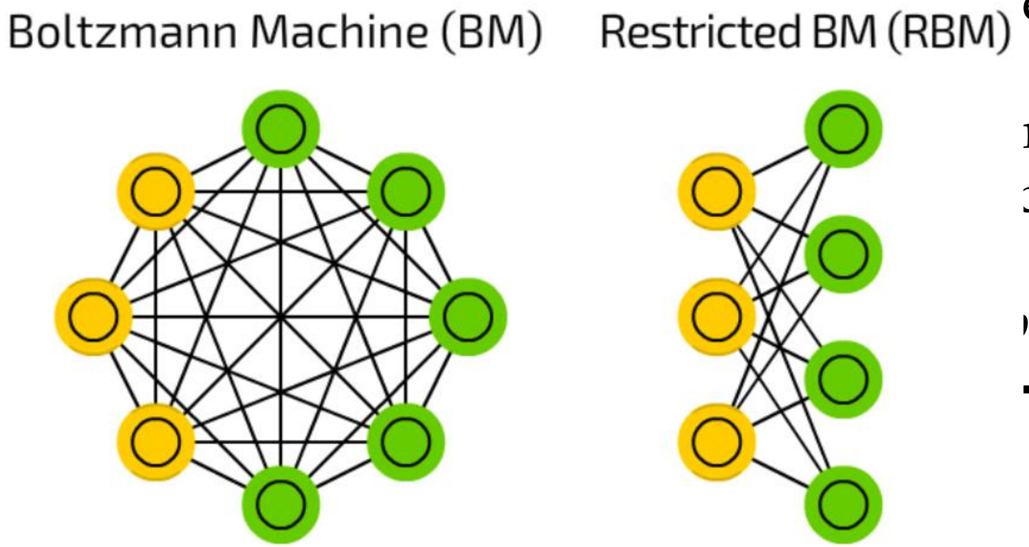
Conditional probability distribution
when h, v is fixed

II. Background

Sparse Restricted Boltzmann machines

RBM (Restricted Boltzmann machines)

- Simple architecture
- en variables(h) and visible variables(v)



- r visible variables
  listributions by assuming **independence** between events
  $(v \mid h), p(h \mid v)$is easier to calculate (one value is fixed)

– nt a

$$\frac{}{2\sigma^2}\mathbf{v}^T\mathbf{v} - \frac{}{\sigma^2}\left(\mathbf{c}^T\mathbf{v} + \mathbf{b}^T\mathbf{h} + \mathbf{h}^T W\mathbf{v}\right)$$

probability over h, v

$$p(h_j|\mathbf{v}) = sigmoid(\frac{1}{\sigma^2}(b_j + \mathbf{w}_j^T\mathbf{v}))$$

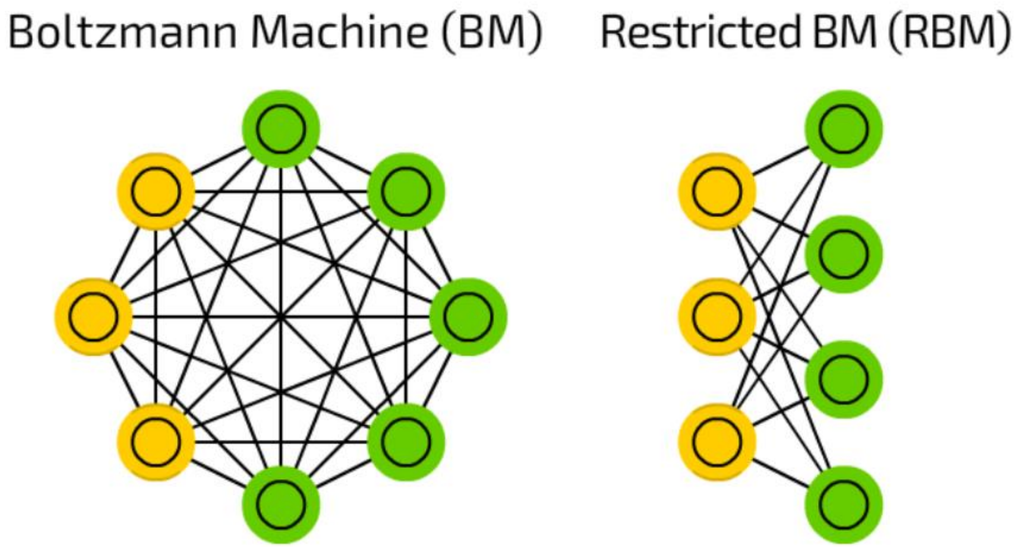
Conditional probability distribution
when h, v is fixed

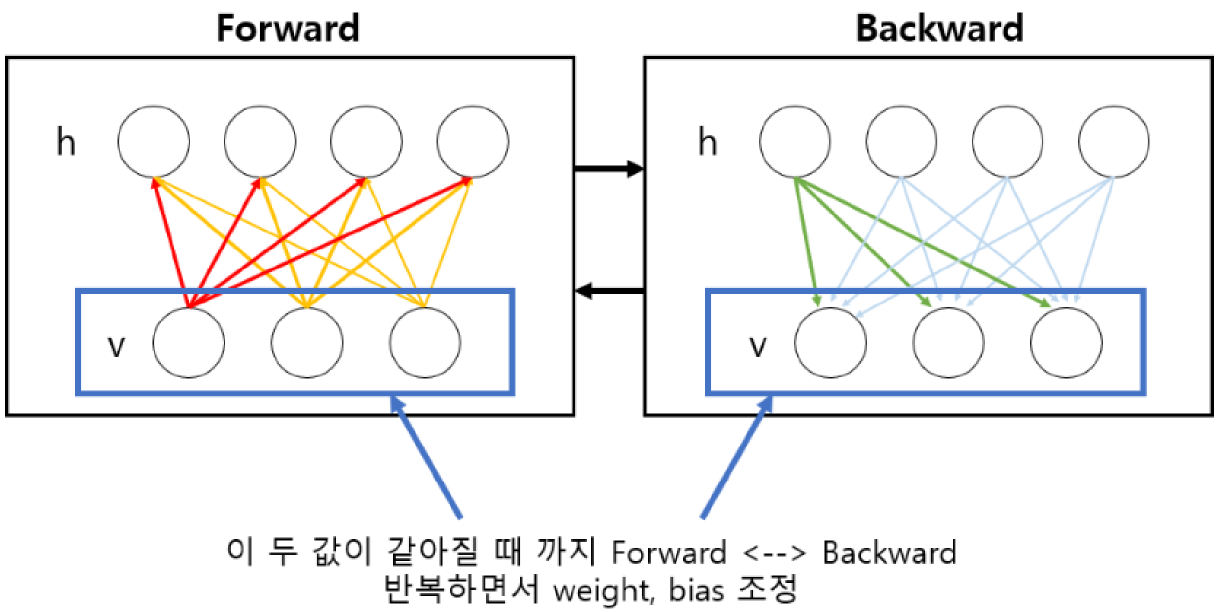
II. Background

Sparse Restricted Boltzmann machines

RBM (Restricted Boltzmann machines)

- Simple architecture





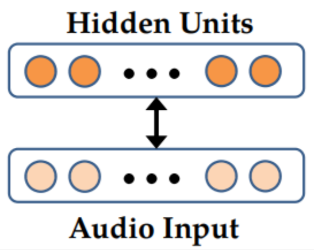
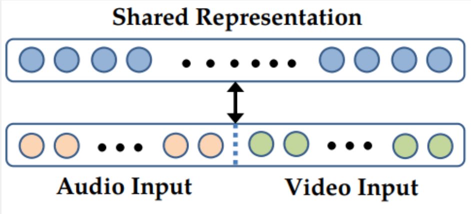
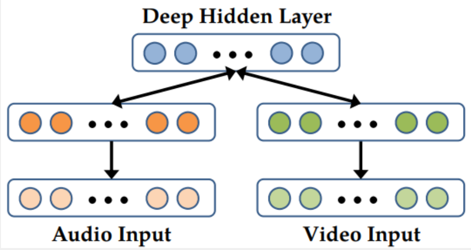
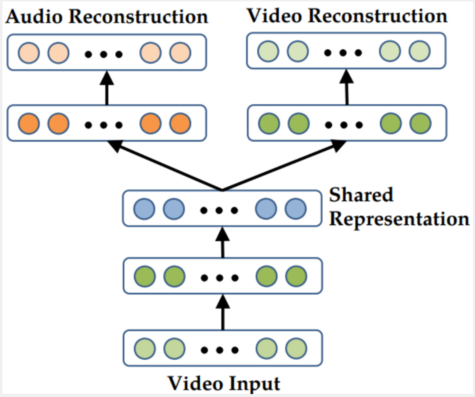
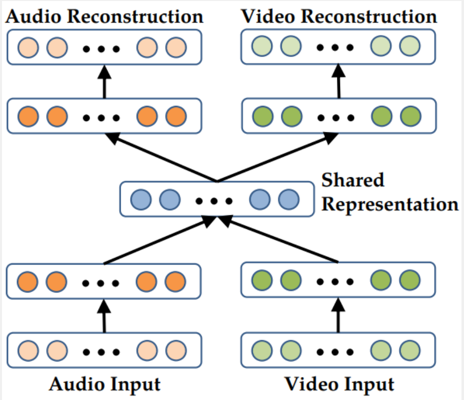
$$\frac{1}{2\sigma^2}\mathbf{v}^T\mathbf{v} - \frac{1}{\sigma^2}\left(\mathbf{c}^T\mathbf{v} + \mathbf{b}^T\mathbf{h} + \mathbf{h}^T W\right.$$

probability over h, v

Conditional probability distribution when h, v is fixed

III. Learning Architectures

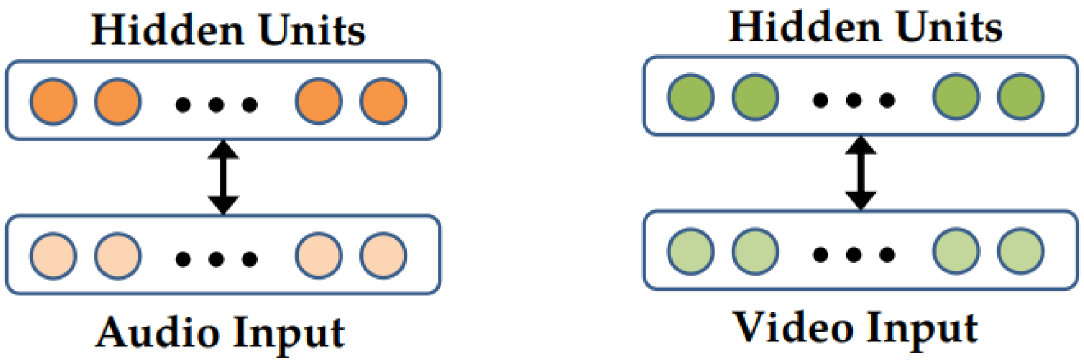
Models Introduction in this paper

<table>
<tr><th>RBM model</th><th>Shallow Bimodal RBM</th><th>Bimodal Deep Belief Network</th><th>Video-Only Deep Autoencoder</th><th>Bimodal Deep Autoencoder</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>

III. Learning Architectures

(1/3) Feature Learning

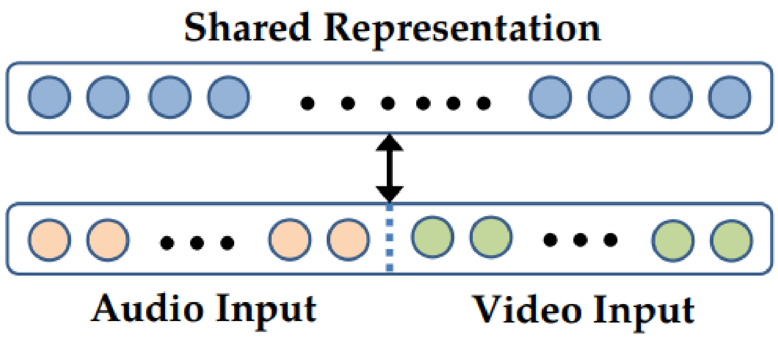
(1,2 / 5) RBM and RBM concatenated



(a) Audio RBM (b) Video RBM

(1) RBM

- Baseline of these experiments
- The posteriors of the hidden variables given the visible variables can be used a new representation for the data.

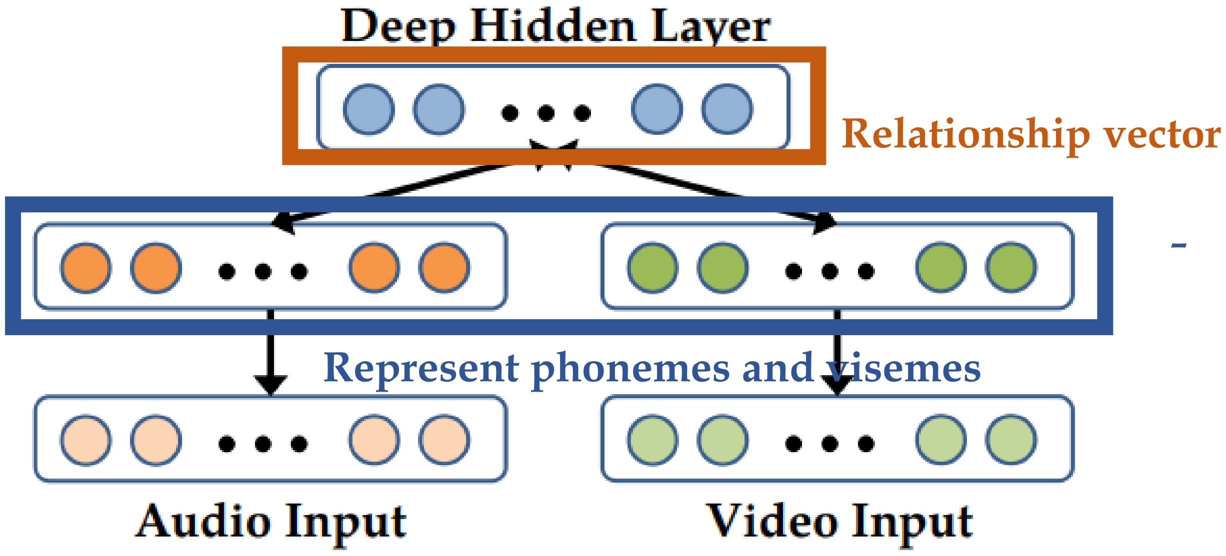


(c) Shallow Bimodal RBM

(2) RBM concatenated

- Jointly models the distribution of the audio and video data (shallow model)
- Because of the highly non-linear correlation, it is <u>hard to learn these correlations</u> and <u>form multimodal representations</u>
(unable to capture correlations <u>across the modalities</u>)

(3 / 5) RBM over the pretrained layers (Bimodal DBN)



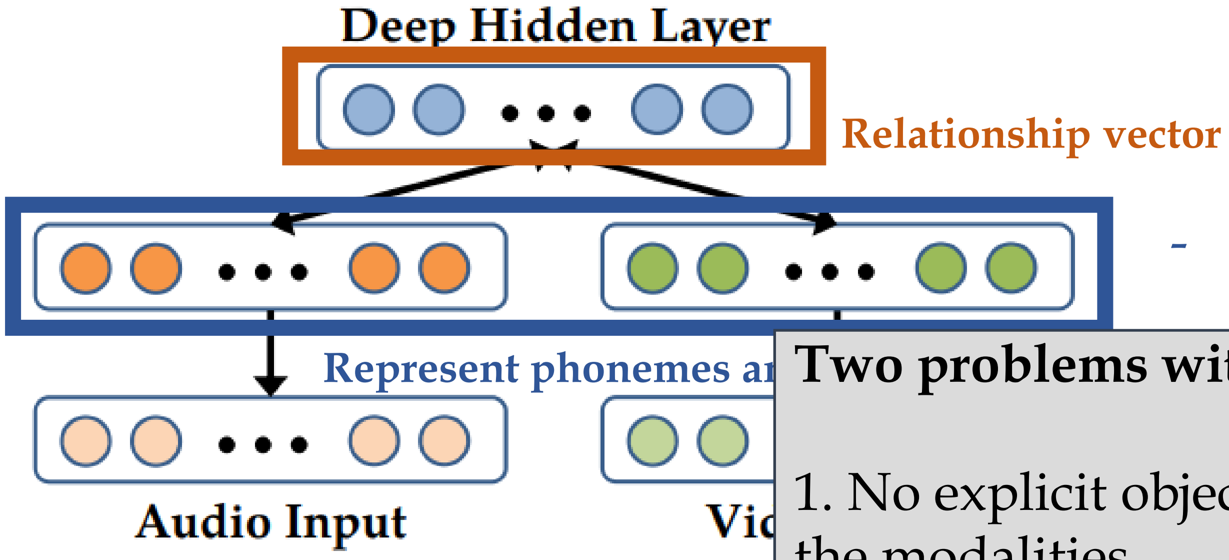
(d) Bimodal DBN

(3) Bimodal DBN

- Represents the data though learned first layer representations
  -> can be easier the model to learn **higher-order correlations across modalities**

- **First layer** : represents phonemes and visemes
  Second layer : relationships between them

(3 / 5) RBM over the pretrained layers (Bimodal DBN)



(d) Bimodal DB

(3) Bimodal DBN

- Represents the data though learned first layer representations
  -> can be easier the model to learn **higher-order correlations across modalities**

- **First layer** : represents phonemes and visemes
  Second layer : relationships between them

Two problems with the Bimodal DBN

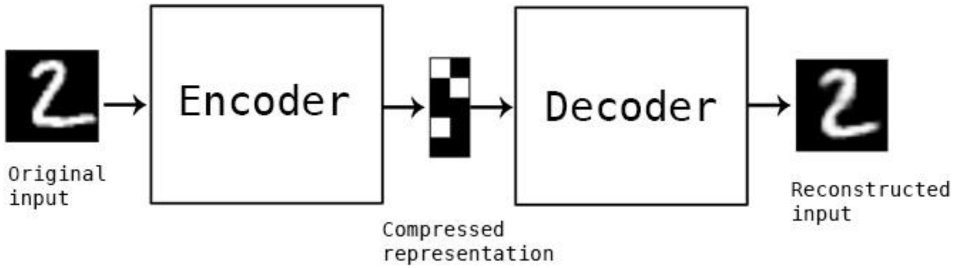
1. No explicit objective for the models to discover correlations across the modalities

2. Models are **clumsy** to use in a cross modality learning setting

=> Deep Autoencoder

(4/5) Deep Autoencoder (Video-only)

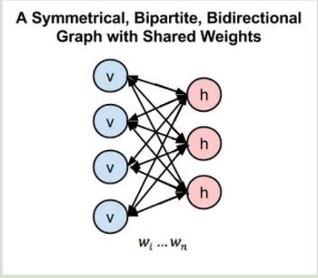
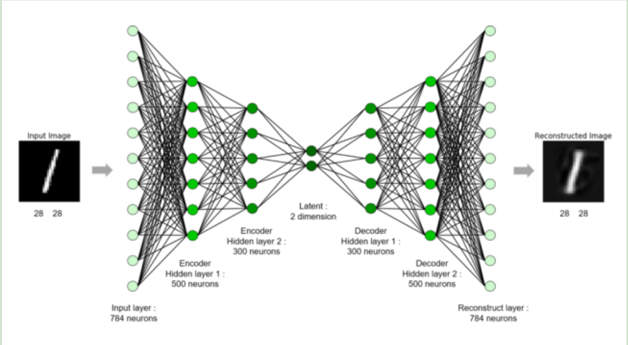
What is the autoencoder?



1. Compress the input data as much as possible (Encoder)

2. Reconstruct the compressed data back to the original input form (Decoder)

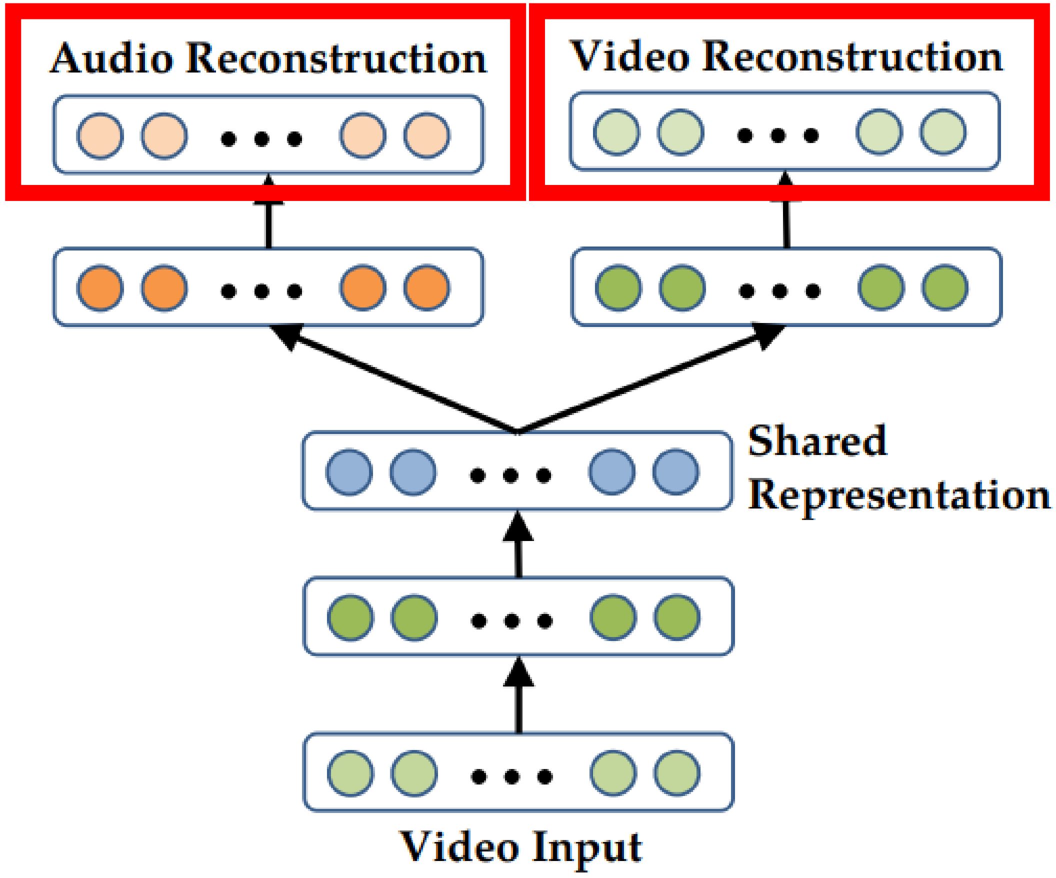
RBM vs Deep autoencoder?

<table>
<tr><th>RBM</th><th>Deep autoencoder</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">Similar to autoencoders, RBMs try to make the reconstructed input from the “hidden” layer as close to the original input as possible.</td></tr>
<tr><td>RBMs use the same matrix for “encoding” and “decoding.”

Trained RBMs can be used as layers in neural networks.</td><td>Encoder matrix and decoder matrix are not same</td></tr>
</table>

III. Learning Architectures

(4/5) Deep Autoencoder (Video-only)



(a) Video-Only Deep Autoencoder

(4) Video-only Deep Autoencoder

- Both modalities during feature learning
  Single modality during supervised training & testing

- Reconstruct <u>Audio & Video representation</u> vectors given only **video as the input**
  (can discover <u>**correlations**</u> across the modalities)

- When multiple modalities are available for the task, it is **less clear** how to use this model as one would need to <u>train a deep autoencoder for each modality</u>

III. Learning Architectures

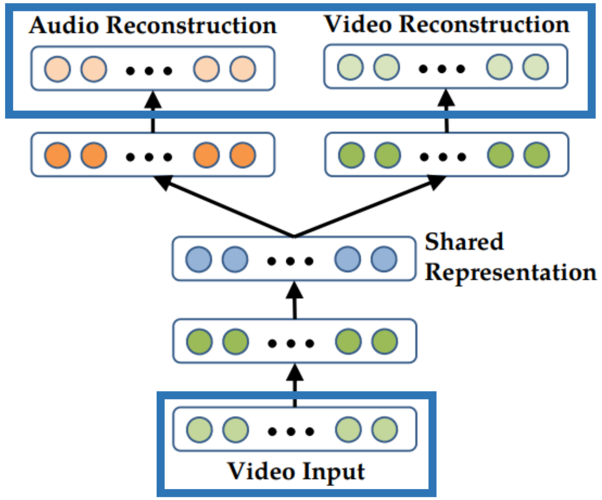
(1/3) Feature Learning

(4/5) Deep Autoencoder (Video-only)

```python
def build_model(self):
    """build (deep) autoencoder model
    """
    # an input placeholder
    input_data = Input(shape=(self.input_dim, ))

    # encoded representation of the input
    encoded = Dense(self.hidden_dim[0], activation='relu')(input_data)
    encoded = Dense(self.hidden_dim[1], activation='relu')(encoded)
    encoded = Dense(self.hidden_dim[2], activation='relu')(encoded)
    # decoded representation of the input
    decoded = Dense(self.hidden_dim[1], activation='relu')(encoded)
    decoded = Dense(self.hidden_dim[0], activation='relu')(decoded)
    decoded = Dense(self.input_dim, activation='sigmoid')(decoded)

    # maps an input to its reconstruction
    self.autoencoder = Model(inputs=input_data, outputs=decoded)
    # maps an input to its encoded representation
    self.encoder = Model(inputs=input_data, outputs=encoded)

    # an encoded input placeholder
    encoded_input = Input(shape=(self.hidden_dim[2], ))
    # retrieve layer of the autoencoder model
    decoder_layer1 = self.autoencoder.layers[-3]
    decoder_layer2 = self.autoencoder.layers[-2]
    decoder_layer3 = self.autoencoder.layers[-1]
    # maps the encoded representation to the input
    self.decoder = Model(encoded_input, decoder_layer3(decoder_layer2(decoder_layer1(encoded_input))))

    # configure the model
    self.autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

Compare input vector and decoded vector using dense layer

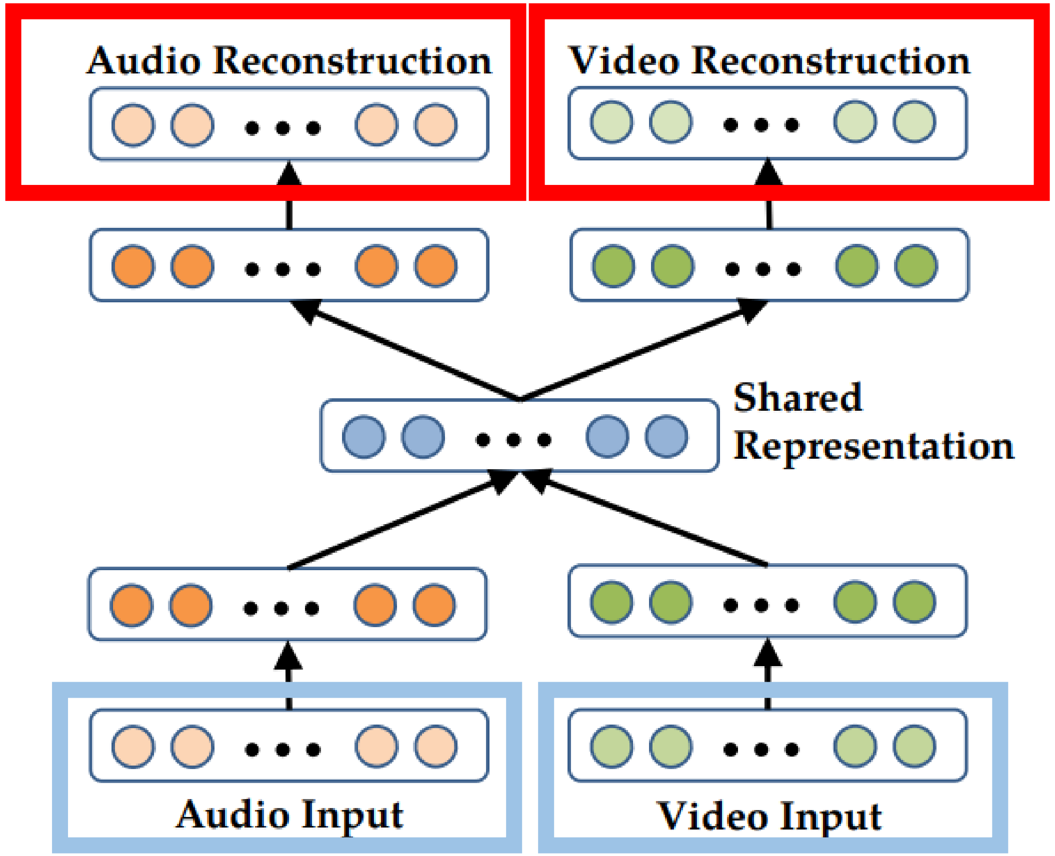
```python
def train_model(self, X_train):
    """train (deep) autoencoder model and save to external file
    """
    self.autoencoder.fit(X_train, X_train,
                         epochs=self.epochs,
                         batch_size=self.batch_size,
                         shuffle=True)
    self.save_model()
```



(a) Video-Only Deep Autoencoder

III. Learning Architectures

(5 / 5) Bimodal Deep Autoencoder



(b) Bimodal Deep Autoencoder

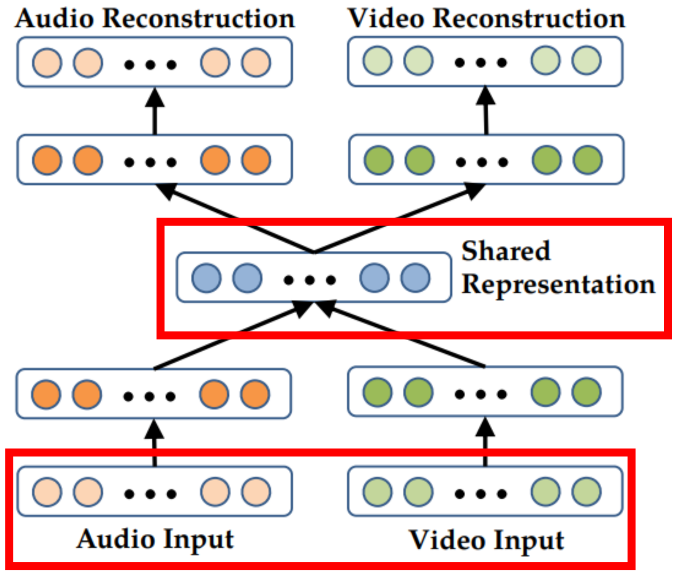
- **Bimodal deep autoencoder** in a denoising fashion
- Use an augmented but noisy dataset with additional examples that have only a single-modality as input
- Video(zero values) + audio(original values)
  => reconstruct both modalities(audio and video)

(5 / 5) Bimodal Deep Autoencoder

```python
def build_model(self):
    """build bimodal (deep) autoencoder model
    """
    input_data_A = Input(shape=(self.input_dim_A, ), name='input_A')
    input_data_V = Input(shape=(self.input_dim_V, ), name='input_V')
    # encoded_input = Input(shape=(self.hidden_dim_shared, ), name='shared_repres')

    encoded_A = Dense(self.hidden_dim_A[0], activation='relu', name='encoded_A_1')(input_data_A)
    encoded_V = Dense(self.hidden_dim_V[0], activation='relu', name='encoded_V_1')(input_data_V)

    encoded_A = Dense(self.hidden_dim_A[1], activation='relu', name='encoded_A_2')(encoded_A)
    encoded_V = Dense(self.hidden_dim_V[1], activation='relu', name='encoded_V_2')(encoded_V)

    shared = Concatenate(axis=1, name='concat')([encoded_A, encoded_V])
    encoded = Dense(self.hidden_dim_shared, activation='relu', name='shared_layer')(shared)

    decoded_A = Dense(self.hidden_dim_A[1], activation='relu', name='decoded_A_2')(encoded)
    decoded_V = Dense(self.hidden_dim_V[1], activation='relu', name='decoded_V_2')(encoded)

    decoded_A = Dense(self.hidden_dim_A[0], activation='relu', name='decoded_A_1')(decoded_A)
    decoded_V = Dense(self.hidden_dim_V[0], activation='relu', name='decoded_V_1')(decoded_V)

    decoded_A = Dense(self.input_dim_A, activation='sigmoid', name='decoded_A')(decoded_A)
    decoded_V = Dense(self.input_dim_V, activation='sigmoid', name='decoded_V')(decoded_V)

    self.autoencoder = Model(inputs=[input_data_A, input_data_V], outputs=[decoded_A, decoded_V])
    self.encoder = Model(inputs=[input_data_A, input_data_V], outputs=encoded)

    self.autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

Compare input vector and decoded vector using dense layer

```python
def train_model(self, X_train_A, X_train_V):
    """train bimodal (deep) autoencoder model and save to external file
    """
    self.autoencoder.fit([X_train_A, X_train_V],
                         [X_train_A, X_train_V],
                         epochs=self.epochs,
                         batch_size=self.batch_size,
                         shuffle=True)
    self.save_model()
```



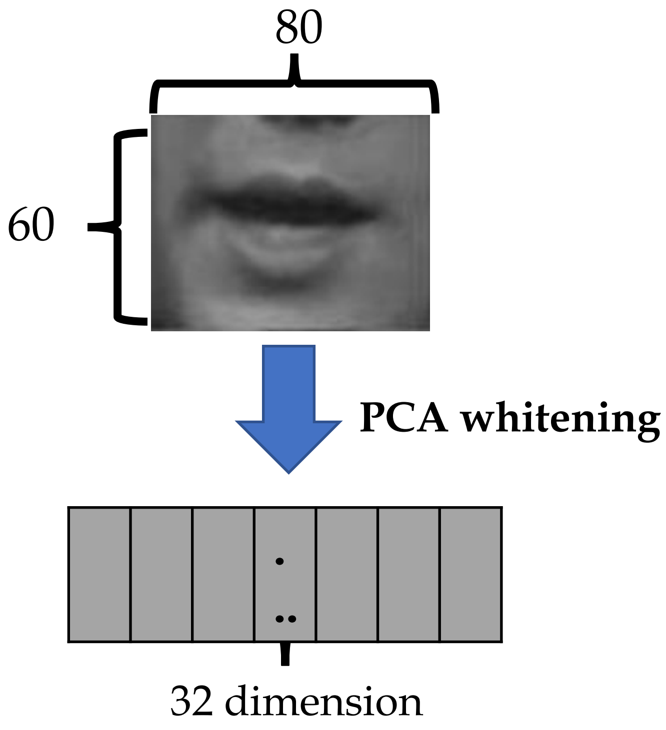
(b) Bimodal Deep Autoencoder

IV. Experiments and Results

1. Data Preprocessing

- Evaluate methods on audio-visual speech classification of isolated letters and digits
- ρ : parameter chosen using cross-validation

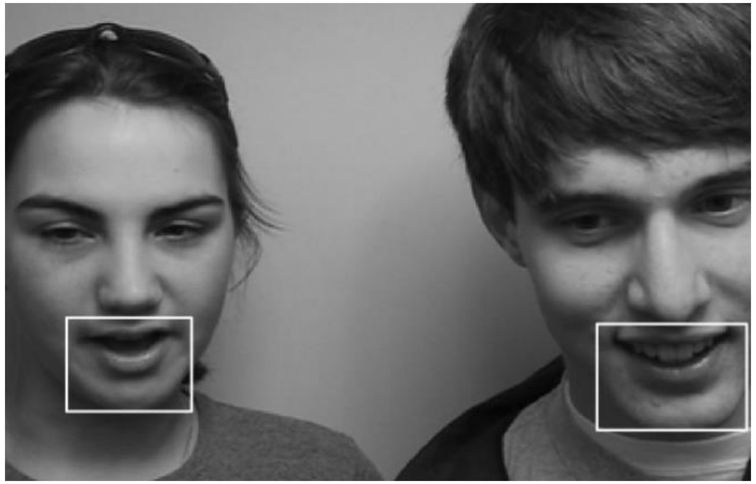
Video dataset preprocessing



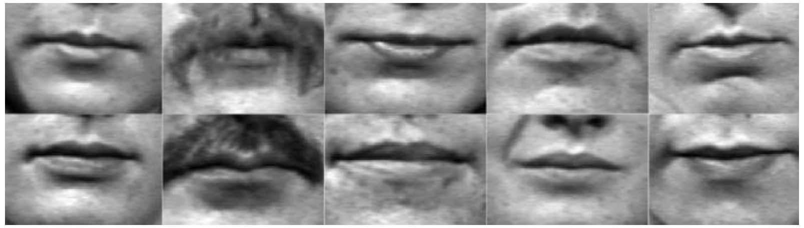
IV. Experiments and Results

2. Datasets used

Datasets	Details
CUAVE (video and audio)	- 36 speakers saying 0 to 9, 5 times - Odd-numbered : test set - Even-numbered : training set
AVLetters (video)	- 10 speakers saying A to Z - **60 x 80 pixels** - **Visual-only** lipreading task
AVLetters2	- 5 speakers saying A to Z, 7 times - New high-definition version of AVLetters
Stanford Dataset	- 23 volunteers saying 0 to 9, A to Z, selected sentences from TIMIT datasets
TIMIT	- Unsupervised audio feature pre-training



CUAVE dataset



AVLetters dataset