Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition

Presented by Peidong Wang

09/09/2016

Yanmin Qian, et al. “Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition.” *IEEE Transactions on Audio, Speech, and Language Processing*. Accepted for publication for a future issue.

Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

Abstract

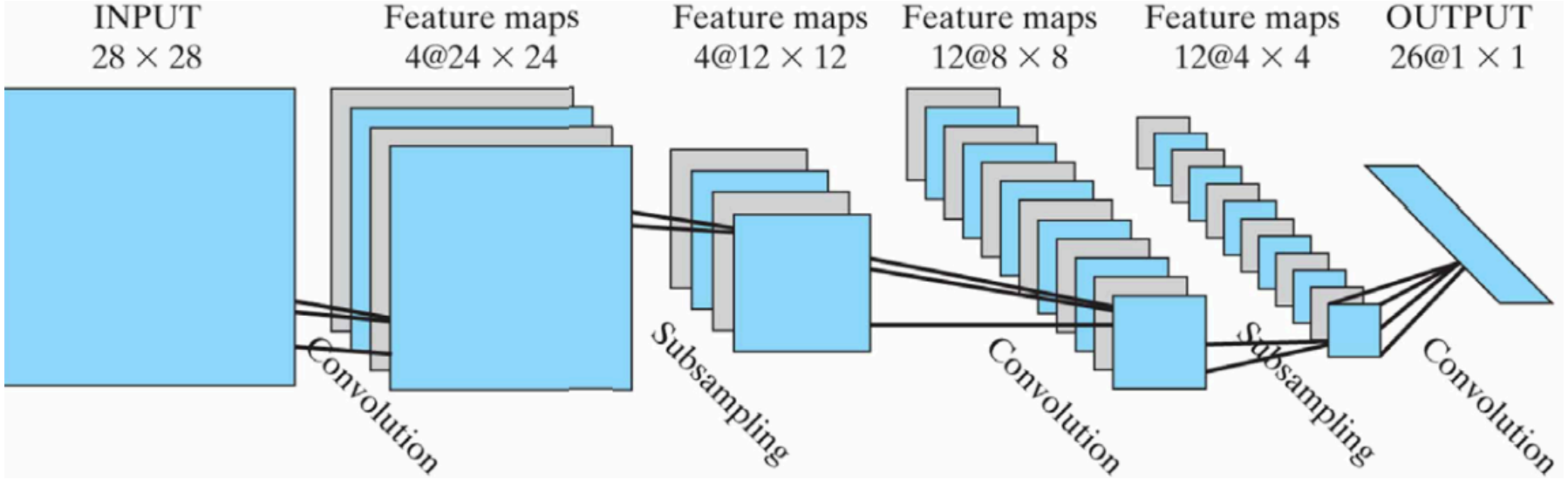
- **ASR:** Previous attempts increasing the number of CNN layers from 2 to 3 gave a degradation.
- **CV:** Recent work in image shows that the accuracy of image classification can be improved by increasing the number of convolutional layers with carefully tuned architecture.
- **ASR:** Very Deep Convolutional Neural Networks uses up to 10 convolutional layers and gets a WER of 8.81% on Aurora4, which is the best published result.

Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

Review of Convolutional Neural Networks

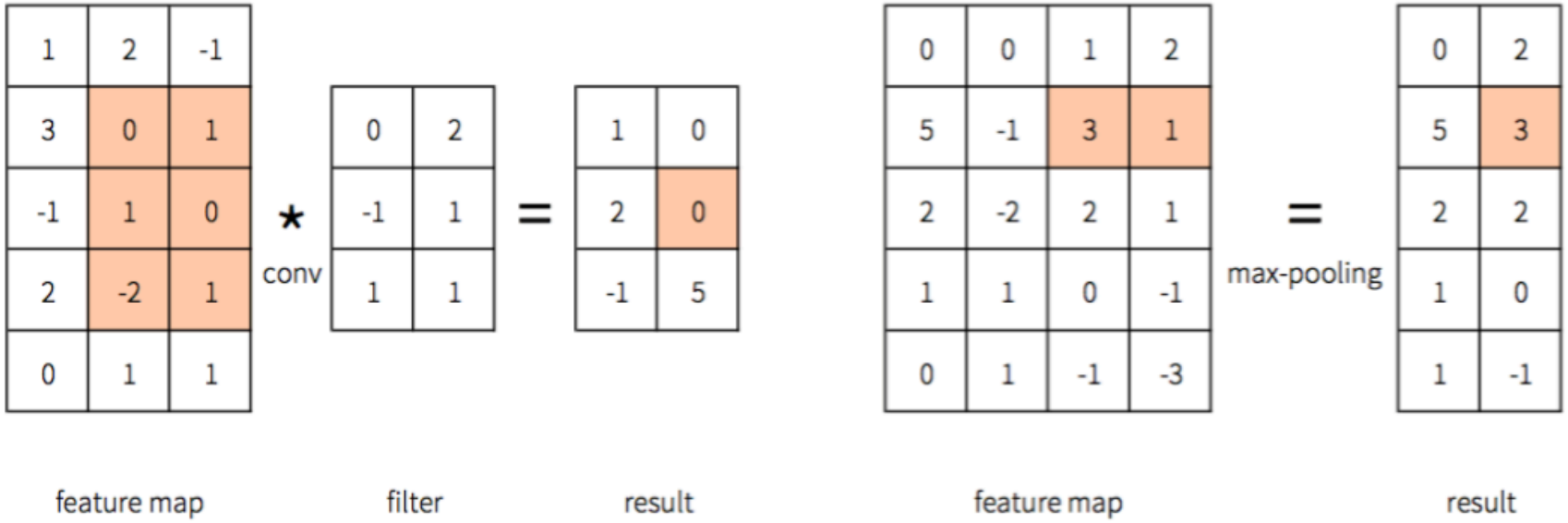
- **A Conventional Convolutional Neural Network (CNN)**



From: Slides in CSE5526 Neural Networks

Review of Convolutional Neural Networks

- **Convolution and Pooling (Subsampling)**



Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

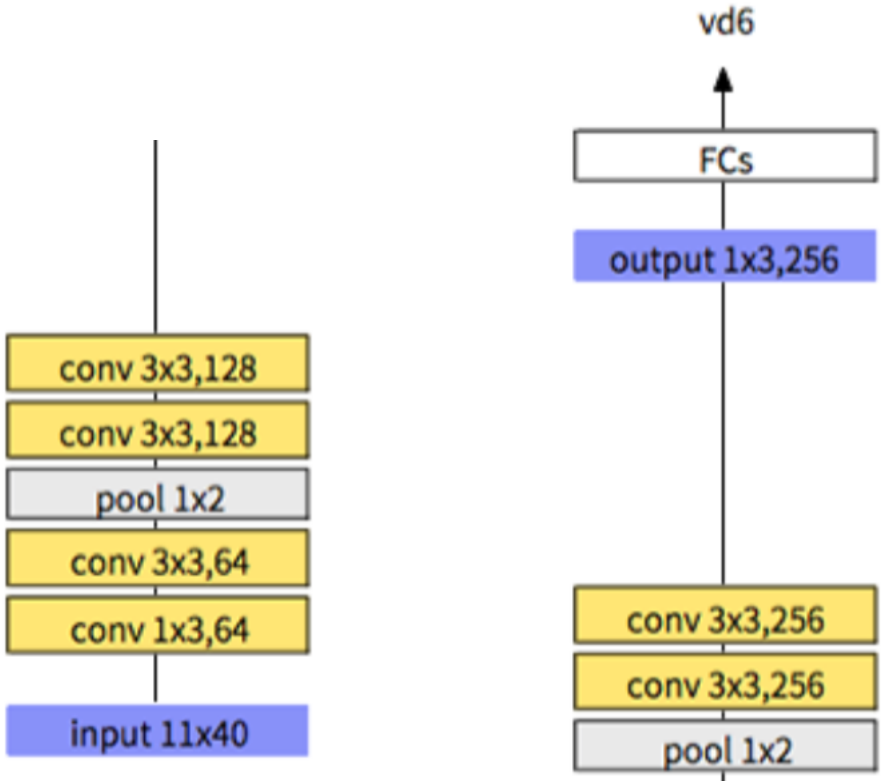
Model Description

- **Context Window Extension**
- A typical size of input features in speech recognition is 11 x 40, where 11 denotes the number of frames in a window, 40 denotes the dimension of FBank features. [*]
- Using this context window size, convolutions can be performed in time 5 times with a filter size of 3, as in the following figure (vd6).

[*] added by the presenter

Model Description

- **Context Window Extension (cont'd)**

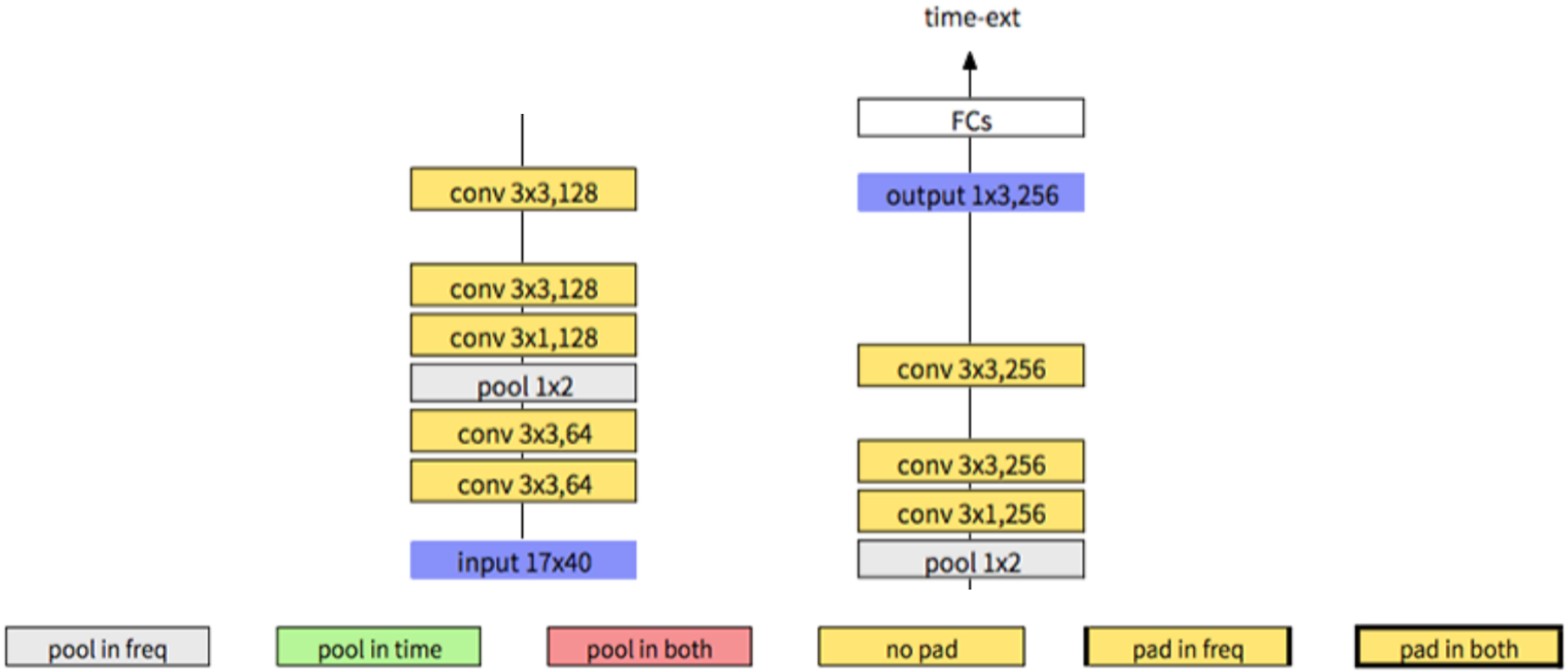


Model Description

- **Context Window Extension (cont'd)**
- In Very Deep Convolutional Neural Networks (VDCNNs), the context window size is extended to 17 (and further to 21), which allows 8 (and 10) convolutions to be performed in time, respectively.

Model Description

- **Context Window Extension (cont'd)**



Model Description

- **Context Window Extension (cont'd)**

TABLE II: WER (%) comparisons of the models having various context window and feature dimension extensions. **F** indicates the size on **F**requency axis and **T** indicates the size on **T**ime axis. **L** indicates the number of convolutional layers in the model.

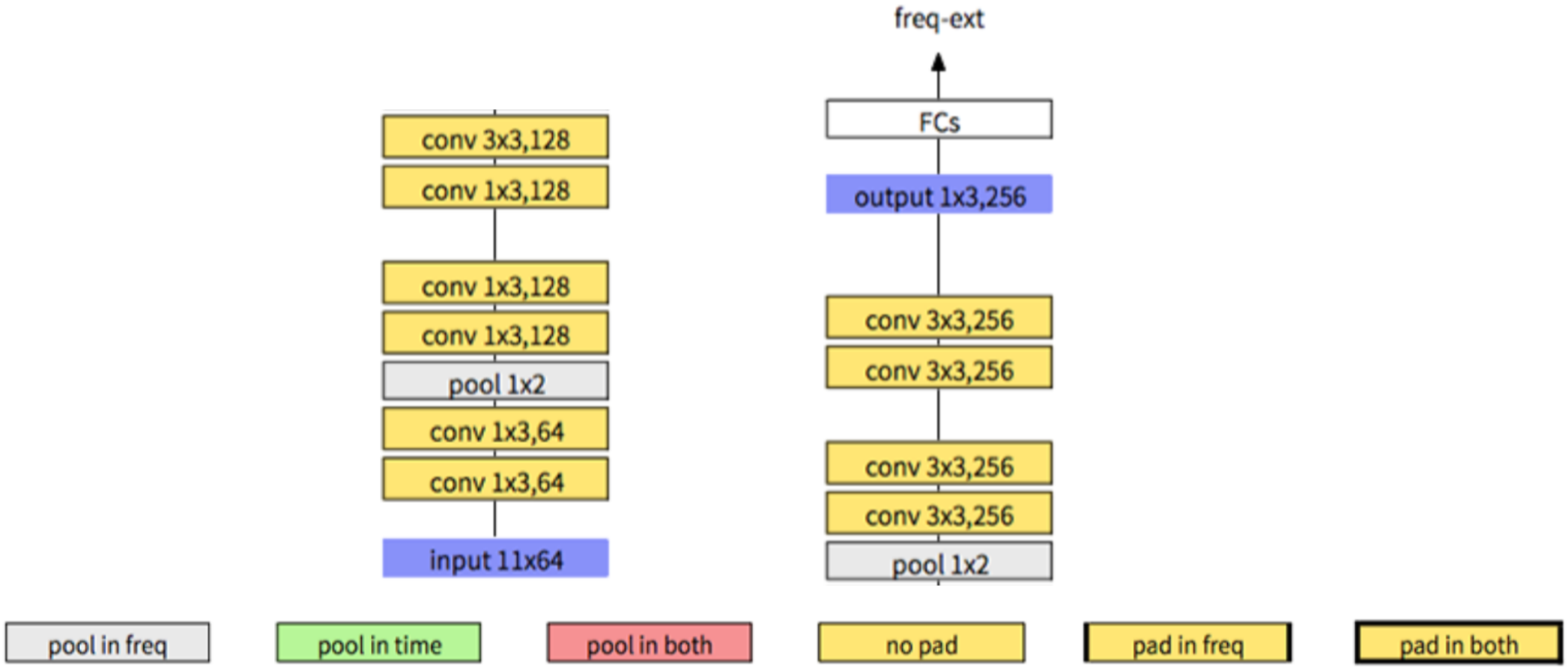
Model	T × F	L	A	B	C	D	AVG
CNN	11 × 40	2	4.11	7.00	6.33	16.09	10.64
vd6	11 × 40	6	3.94	6.86	6.33	15.56	10.34
time-ext	17 × 40	8	3.72	6.57	5.83	14.79	9.84

Model Description

- **Feature Dimension Extension**
- Based on 40-dim FBank features, at most 6 convolutions and 2 poolings can be performed in frequency, leading to the vd6 model.
- In VDCNN, the FBank features are extended to 64-dim, so that 4 more convolutions can be performed in frequency.

Model Description

- **Feature Dimension Extension (cont'd)**

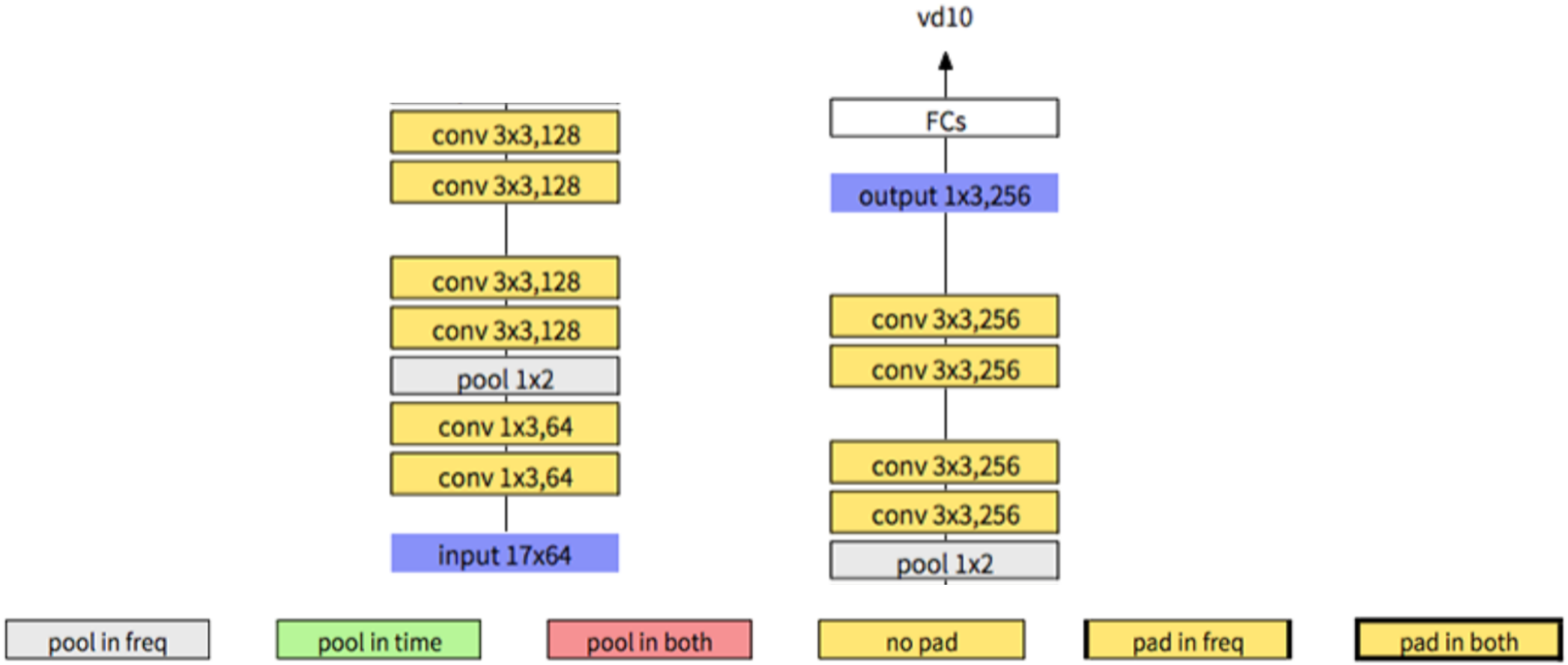


Model Description

- **Feature Dimension Extension (cont'd)**
- Finally the input extension is performed in both time and frequency, leading to a 17 x 64 input. The resulting model is named vd10.

Model Description

- **Feature Dimension Extension (cont'd)**

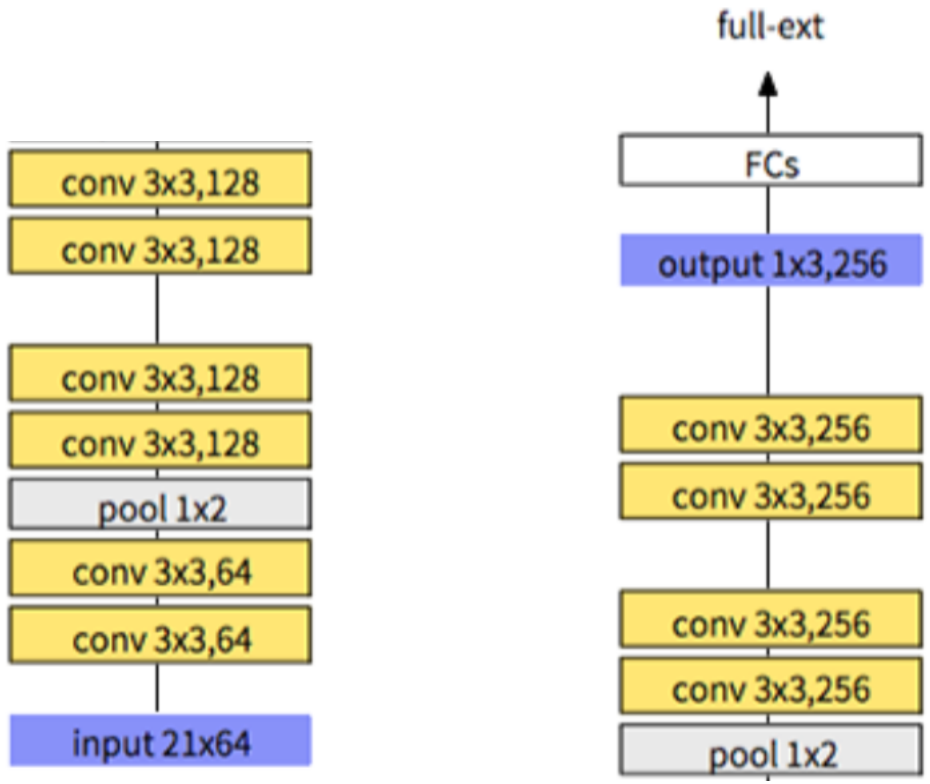


Model Description

- **Feature Dimension Extension (cont'd)**
- The full-ext model further extends the number of time frames to 21 so that 2 more convolution operations can be performed in time, giving 10 convolution operations in both time and frequency.

Model Description

- **Feature Dimension Extension (cont'd)**



Model Description

- **Feature Dimension Extension (cont'd)**
- To confirm that the performance gain is not from the extended input features, a model with the same wider input features (17 x 64) but shallow convolutional layers is developed.

Model Description

- **Feature Dimension Extension (cont'd)**

TABLE II: WER (%) comparisons of the models having various context window and feature dimension extensions. **F** indicates the size on **F**requency axis and **T** indicates the size on **T**ime axis. **L** indicates the number of convolutional layers in the model.

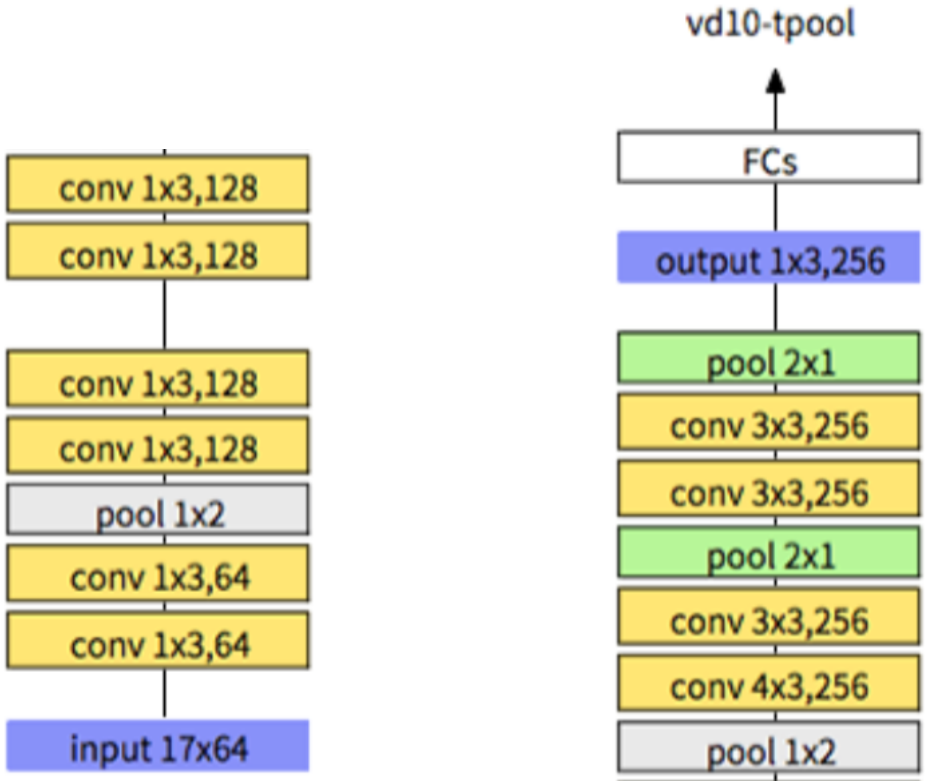
Model	T × F	L	A	B	C	D	AVG
CNN	11 × 40	2	4.11	7.00	6.33	16.09	10.64
vd6	11 × 40	6	3.94	6.86	6.33	15.56	10.34
time-ext	17 × 40	8	3.72	6.57	5.83	14.79	9.84
freq-ext	11 × 64	10	3.79	6.51	6.26	15.19	10.02
vd10	17 × 64	10	4.13	6.62	5.92	14.53	9.78
full-ext	21 × 64	10	4.04	6.23	5.40	13.86	**9.28**
CNN2	17 × 64	2	4.20	7.36	6.84	16.36	10.96

Model Description

- **Pooling in Time**
- You may have noticed that the VDCNN models all use pooling in frequency and do no pooling in time.
- To investigate whether pooling in time is helpful, vd10-tpool is designed.

Model Description

- **Pooling in Time (cont'd)**



pool in freq | pool in time | pool in both | no pad | pad in freq | pad in both

Model Description

- **Pooling in Time (cont'd)**

TABLE III: WER (%) comparison of the models with or without pooling in time

Model	A	B	C	D	AVG
vd10	4.13	6.62	5.92	14.53	9.78
vd10-tpool	3.68	6.46	6.13	15.03	9.91

Model Description

- **Padding in Feature Maps**
- In most work on CNNs for speech recognition, the convolutions are performed without padding.
- Padding can save the size of feature maps and better utilize the border information.

Model Description

- **Padding in Feature Maps (cont'd)**

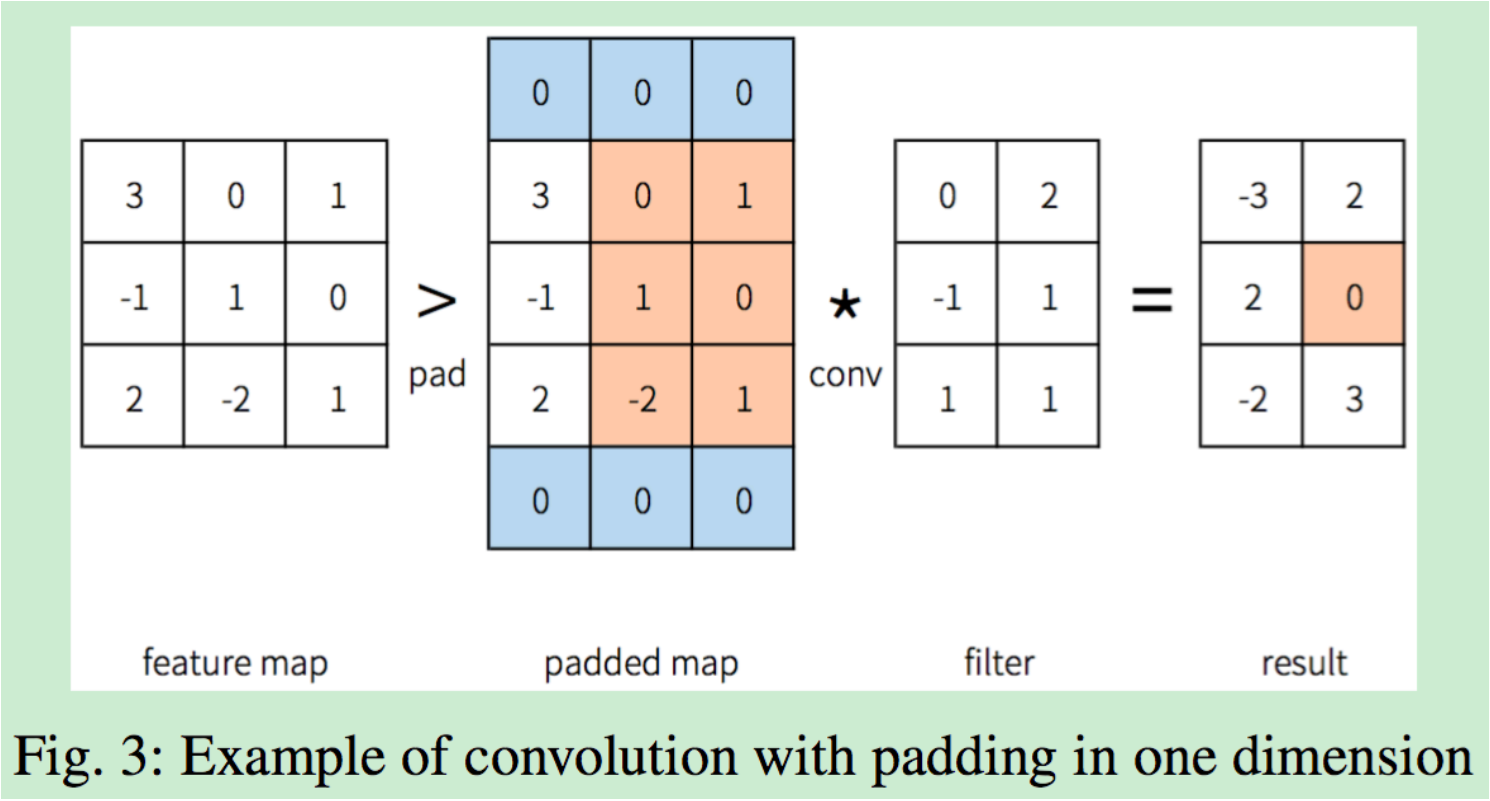


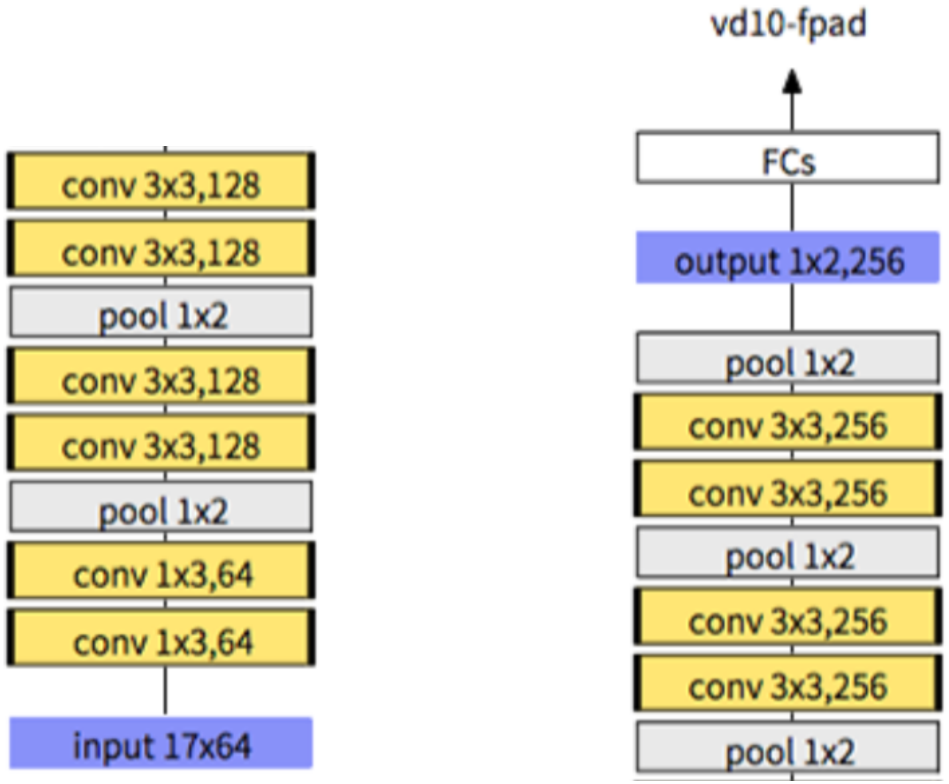
Fig. 3: Example of convolution with padding in one dimension

Model Description

- **Padding in Feature Maps (cont'd)**
- Model vd10-fpad pads only in frequency, allowing more pooling operations in frequency.

Model Description

- **Padding in Feature Maps (cont'd)**

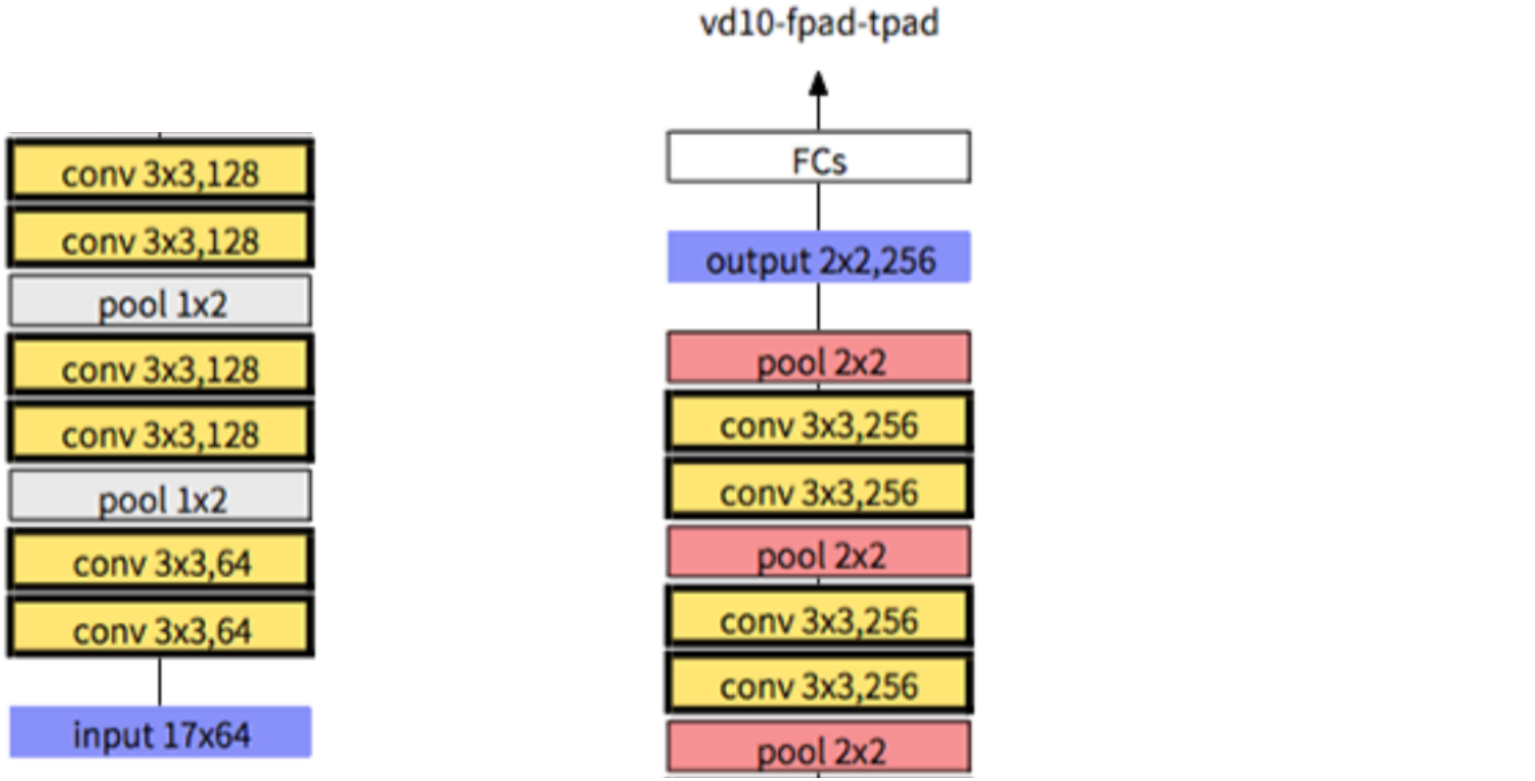


Model Description

- **Padding in Feature Maps (cont'd)**
- Padding in both dimensions is also applied, which is indicated as vd10-fpad-tpad.
- In this model, considering that pooling is a necessary approach to reduce the feature map size, pooling in time is also applied.

Model Description

- **Padding in Feature Maps (cont'd)**



Model Description

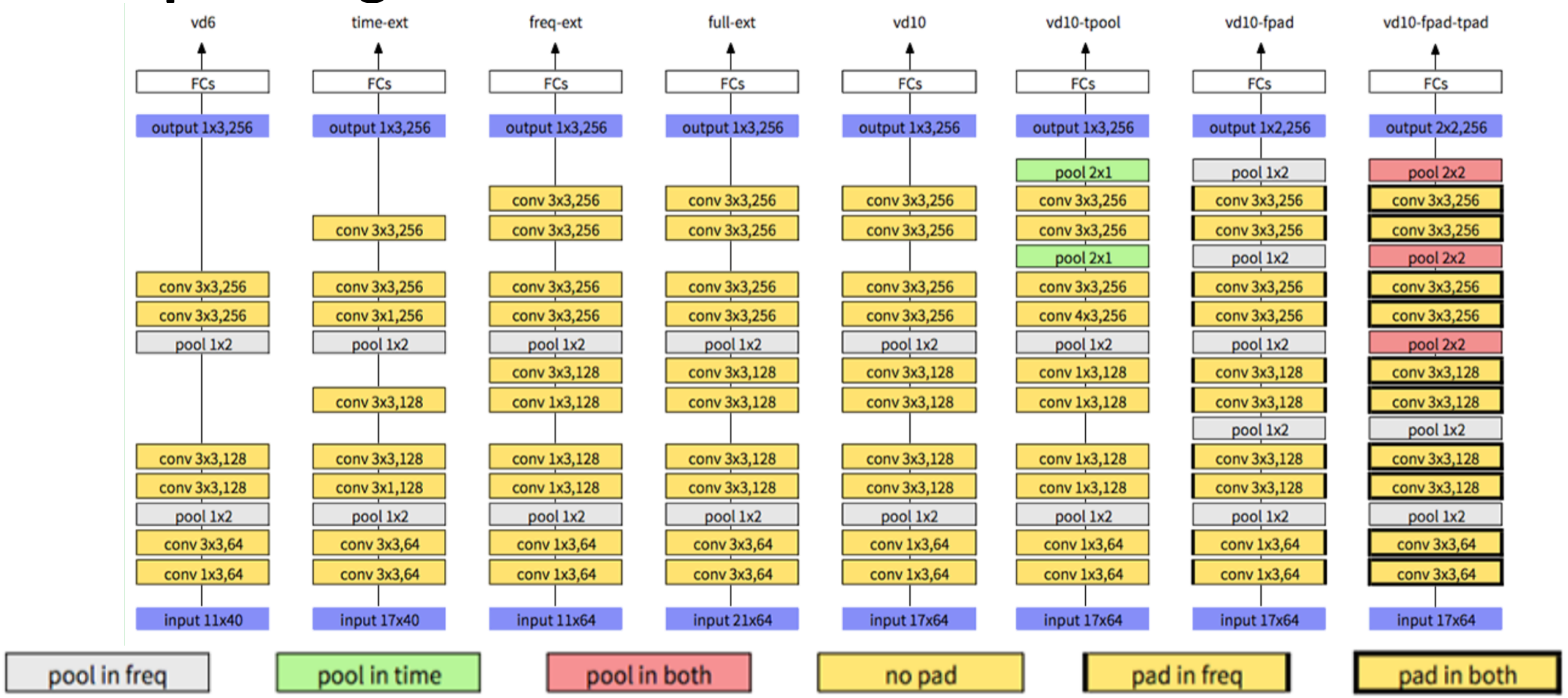
- **Padding in Feature Maps (cont'd)**

TABLE IV: WER (%) comparison of the models with different padding strategies

Model	A	B	C	D	AVG
CNN	4.11	7.00	6.33	16.09	10.64
vd10	4.13	6.62	5.92	14.53	9.78
vd10-fpad	3.57	6.17	5.31	14.24	9.38
vd10-fpad-tpad	3.27	5.61	5.32	13.52	**8.81**

Model Description

- **Complete Figure**



Model Description

- **Complete Figure (cont'd)**

model	vd6	time-ext	freq-ext	vd10	full-ext	CNN	CNN2
input map size	11 × 40	17 × 40	11 × 64	17 × 64	21 × 64	11 × 40	17 × 64
#(conv. layers)	6	8	10	10	10	2	2
64 feature maps	1 × 3 3 × 3 [1 × 2]	3 × 3 3 × 3 [1 × 2]	1 × 3 1 × 3 [1 × 2]	1 × 3 1 × 3 [1 × 2]	3 × 3 3 × 3 [1 × 2]	—	—
128 feature maps	3 × 3 3 × 3 [1 × 2]	3 × 1 3 × 3 3 × 3 [1 × 2]	1 × 3 1 × 3 1 × 3 3 × 3 [1 × 2]	3 × 3 3 × 3 3 × 3 3 × 3 [1 × 2]	3 × 3 3 × 3 3 × 3 3 × 3 [1 × 2]	—	—
256 feature maps	3 × 3 3 × 3	3 × 1 3 × 3 3 × 3	3 × 3 3 × 3 3 × 3 3 × 3	3 × 3 3 × 3 3 × 3 3 × 3	3 × 3 3 × 3 3 × 3 3 × 3	9 × 9 [1 × 3] 3 × 4	13 × 13 [1 × 4] 5 × 6
output map size	1 × 3					1 × 8	

Model Description

- **1 Channel vs. 3 Channels Based Input Feature Maps**
- VDCNNs use one channel feature map as input, i.e. the static FBank feature.
- Most work in speech recognition, however, uses three-channel features (static, Δ, and ΔΔ).
- The number of input channels are compared for VDCNN.

Model Description

- **1 Channel vs. 3 Channels Based Input Feature Maps (cont'd)**

TABLE V: WER (%) comparison of the models using one channel or three channels as inputs. # indicates the number of input feature channels.

Model	#	A	B	C	D	AVG
vd6	1	3.94	6.86	6.33	15.56	10.34
	3	4.13	7.14	6.16	15.60	10.48
vd10	1	4.13	6.62	5.92	14.53	9.78
	3	3.90	6.93	6.26	14.75	10.01
vd10-fpad-tpad	1	3.27	5.61	5.32	13.52	8.81
	3	3.79	6.11	5.60	13.62	9.13

Model Description

- **1 Channel vs. 3 Channels Based Input Feature Maps (cont'd)**
- It is interesting to find that 1 channel base VDCNNs are better than the models using 3 channels.
- One possible explanation would be that the information in the dynamic features may be better extracted from the raw static features directly by VDCNN.

Model Description

- **1 Channel vs. 3 Channels Based Input Feature Maps (cont'd)**
- Another explanation may be as follows.

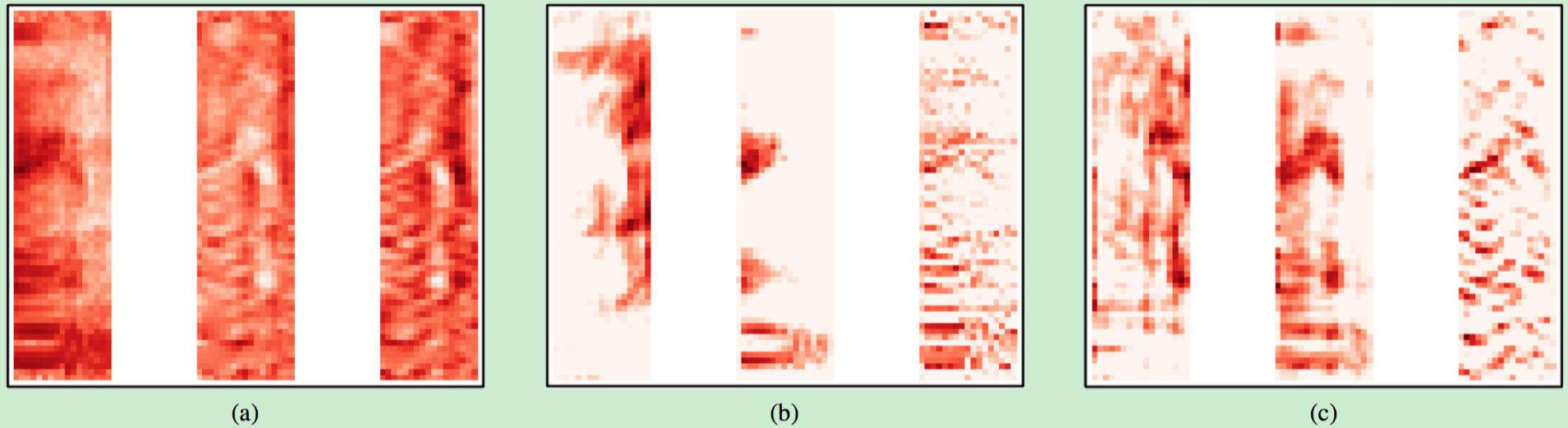


Fig. 4: One speech frame is selected to show the influence of one or three input channels.(a) Illustration of all three channels (i.e. static, Δ and $\Delta\Delta$ features) of the selected frame. (b) Illustration of some feature maps of the first convolutional layer, using one channel as input. (c) Illustration of some feature maps of the first convolutional layer, using three channels as inputs.

Model Description

- **Model Parameter Size**
- It is observed that although the number of convolutional layers is increased significantly in the proposed VDCNN, the total parameter size is smaller than the baseline CNN and DNN.

Model Description

- **Model Parameter Size (cont'd)**

TABLE VI: Comparison of model parameters size

Model	#Params	#Conv.	#Neck	#MLP
DNN	23.67M	-	-	23.67M
CNN	17.62M	0.85M	4.19M	12.58M
vd6	15.29M	1.14M	1.57M	12.58M
vd10	16.74M	2.59M	1.57M	12.58M
vd10-fpad	16.22M	2.59M	1.05M	12.58M
vd10-fpad-tpad	17.30M	2.62M	2.10M	12.58M

Model Description

- **Convergence of Very Deep CNNs**
- The VDCNN converges faster than other model types, in terms of the number of epochs[*].
- Accordingly, although VDCNNs need more computations in each iteration (9.5 times more computations compared to the baseline CNN), the VDCNNs take comparable time for model training.

[*] added by the presenter

Model Description

- **Convergence of Very Deep CNNs (cont'd)**

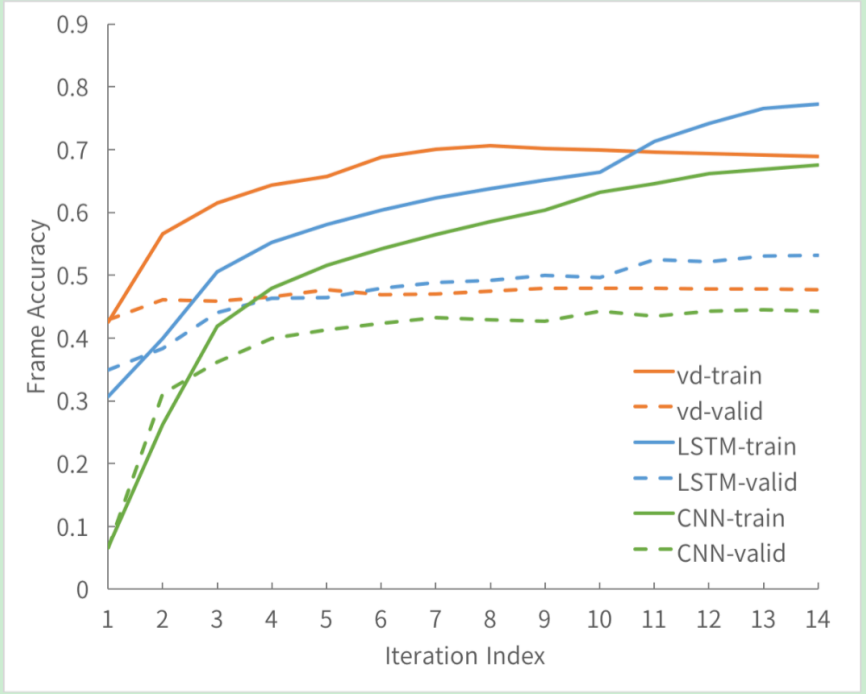


Fig. 5: Convergence curve comparison of different models: LSTM, traditional shallow CNN and the proposed very deep CNN

Model Description

- **Noise Robustness of Very Deep CNNs**

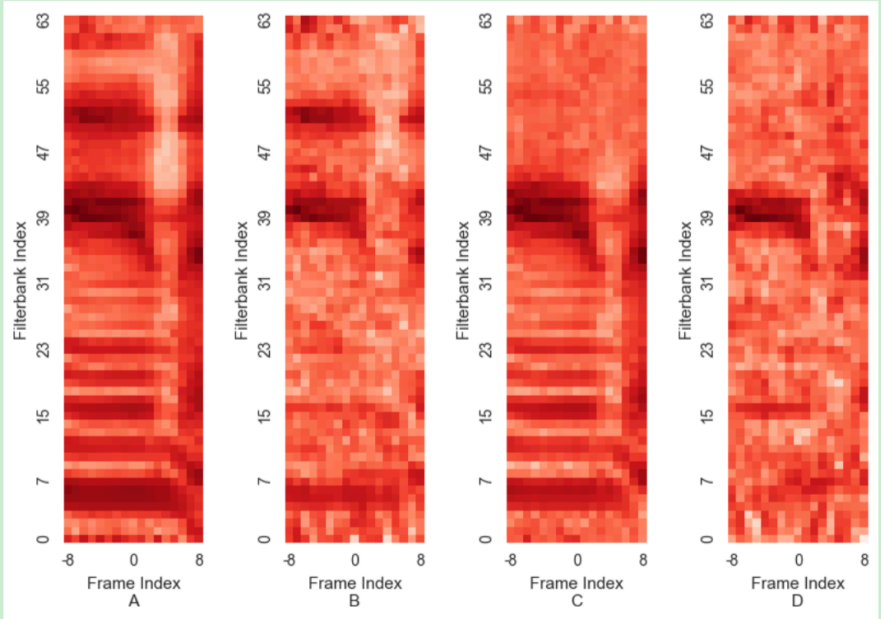


Fig. 6: One speech frame is selected to show the influence of additive noise and channel distortion. Illustration of static features of the same frame in Aurora4 from four conditions A, B, C and D in order.

Model Description

- **Noise Robustness of Very Deep CNNs (cont'd)**
- To better understand how VDCNN processes noisy speech, each condition (A, B, C or D) of this frame is propagated through the best performing model vd10-fpad-tpad.
- The outputs of the 1^{st} convolutional layer and the 6^{th} convolutional layer for A, B, C and D are plotted in the next figures.

Model Description

- **Noise Robustness of Very Deep CNNs (cont'd)**

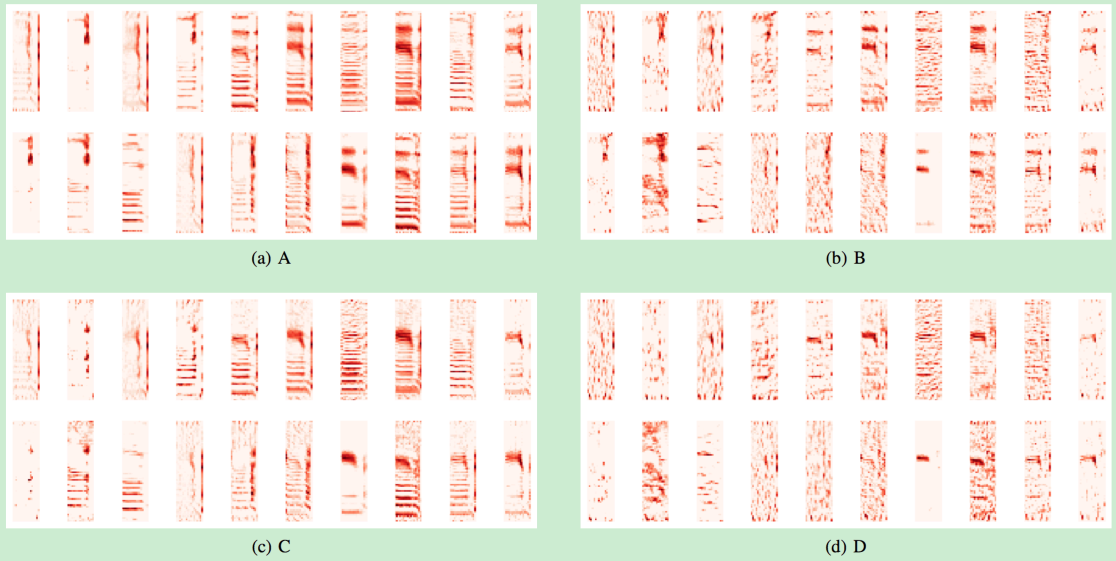


Fig. 7: Selected feature maps of the first convolutional layer using the same single frame from 4 different conditions in Aurora4

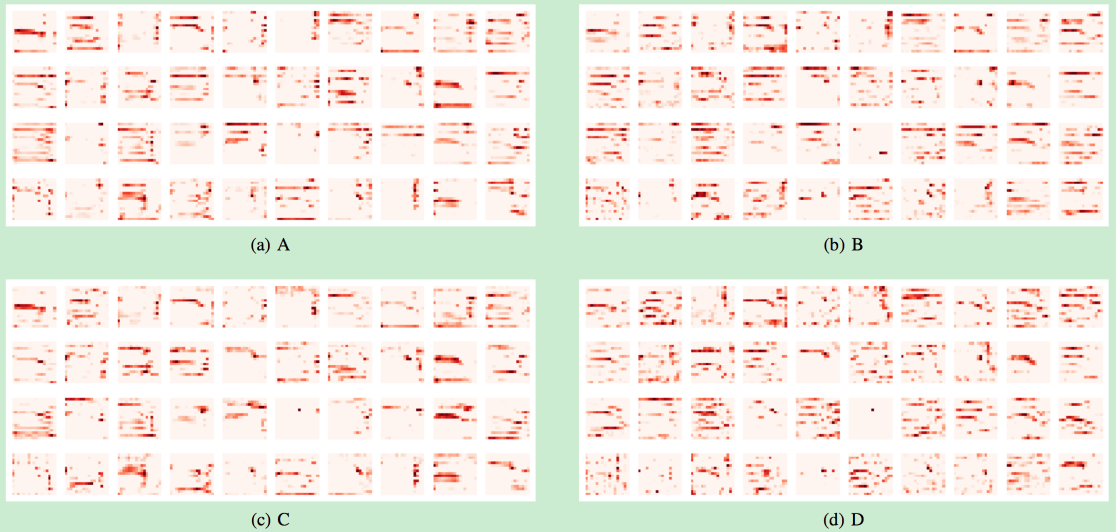


Fig. 8: Selected feature maps of the sixth convolutional layer using the same single frame from 4 different conditions in Aurora4

Model Description

- **Noise Robustness of Very Deep CNNs (cont'd)**
- To further verify the observation, the differences between noisy feature maps and clean feature maps are measured for all convolutional layers.
- Using data in the test, we compute the averaged mean square error (MSE) to evaluate the differences between the three noisy conditions and the clean condition.

Model Description

- **Noise Robustness of Very Deep CNNs (cont'd)**
- The MSE values after all operations are show below.

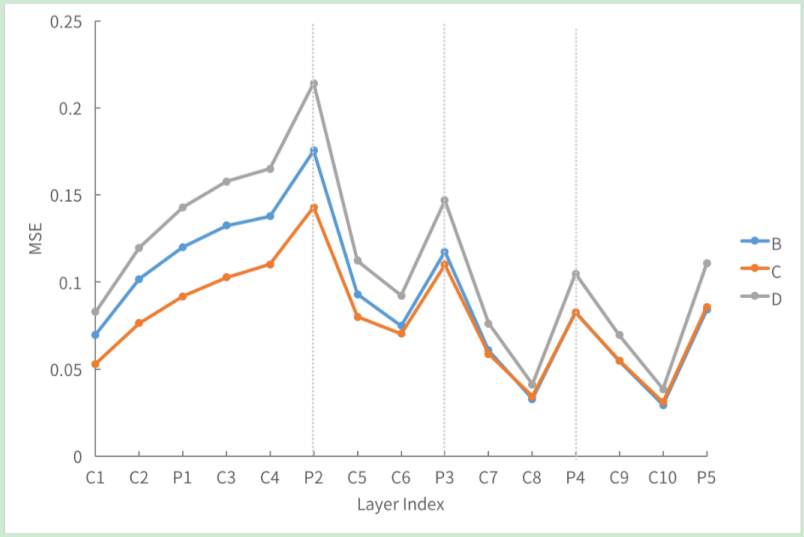


Fig. 9: The Mean Square Error (MSE) variation of different layers of the very deep CNN (`vd10-fpad-tpad` is used) is illustrated. The MSE is calculated between layer outputs using noisy inputs (B, C and D, respectively) and clean inputs (A).

Model Description

- **Noise Robustness of Very Deep CNNs (cont'd)**
- The MSE values for different CNN models.

TABLE VII: The Mean Square Error (MSE) of outputs before the final softmax operation of different models is calculated. The MSE is calculated between the outputs using noisy inputs (B, C and D, respectively) and clean inputs (A).

Model	B	C	D
CNN	3.0282	2.3778	5.1597
vd6	2.9182	2.3515	4.3358
vd10	2.5510	2.0103	4.0397
vd10-fpad	2.2857	2.0037	3.7599
vd10-fpad-tpad	1.7611	1.4873	2.9115

Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

Experiments

- **Experimental Setup**
- The GMM-HMM system is built with Kaldi.
- All neural network models, including DNN/CNN/LSTM, are trained using CNTK.
- The standard testing pipeline in Kaldi recipes are used for decoding and scoring.
- A similar structure (IBM-VGG) designed by researchers in IBM and NYU is also constructed for comparison.

Experiments

- **Evaluation on Aurora4**
- Aurora4 is a medium vocabulary task based on the Wall Street Journal (WSJ0).
- Training sets contain 14276 utterances.
- Four conditions, A, B, C and D, as mentioned before.

Experiments

- **Evaluation on Aurora4 (cont'd)**

TABLE VIII: WER (%) comparison of different models on Aurora4

Model	A	B	C	D	AVG
DNN	4.17	7.46	7.19	16.57	11.11
CNN	4.11	7.00	6.33	16.09	10.64
LSTM	3.92	7.21	6.63	15.94	10.68
vd6	3.94	6.86	6.33	15.56	10.34
vd10	4.13	6.62	5.92	14.53	9.78
vd10-fpad	3.57	6.17	5.31	14.24	9.38
vd10-fpad-tpad	3.27	5.61	5.32	13.52	**8.81**
IBM-VGG	3.92	6.15	5.34	14.20	9.38

Experiments

- **Evaluation on AMI**
- AMI corpus contains around 100 hours of meeting records.
- The signal was captured and synchronized with multiple microphones such as individual head microphones (IHM, close-talk) and microphone arrays (single distant microphone (SDM) and multiple distant microphones (MDM)).
- MDM was processed by a standard beamforming algorithm to generate a single channel dataset.

Experiments

- **Evaluation on AMI (cont'd)**
- The size of input features is investigated.

TABLE IX: WER (%) comparison of the proposed very deep CNNs on AMI MDM condition with different input sizes. **F** indicates the size of the **F**requency axis and **T** indicates the size of the **T**ime axis. **L** indicates the number of convolutional layers in the model.

Model	T × F	L	dev	eval
vd6	11 × 40	6	46.5	51.1
time-ext	17 × 40	8	45.5	50.1
freq-ext	11 × 64	10	45.7	50.7
vd10	17 × 64	10	44.8	49.3
full-ext	21 × 64	10	**44.5**	**49.0**

Experiments

- **Evaluation on AMI (cont'd)**
- The effect of other designs are also investigated.

TABLE X: WER (%) comparison of the proposed very deep CNNs on AMI MDM condition with different pooling and padding strategies. **F** indicates the **F**requency axis and **T** indicates the **T**ime axis. **#** indicates the number of input channels.

Model	#	Pooling	Padding	dev	eval
vd10	3	F	—	45.7	50.5
vd10	1	F	—	44.8	49.3
vd10-tpool		F & T	—	45.0	49.6
vd10-fpad		F	F	43.7	48.2
vd10-fpad-tpad		F & T	F & T	**42.5**	**46.9**

Experiments

- **Evaluation on AMI (cont'd)**
- To better explain the superiority of VDCNNs, we use some related feature maps.

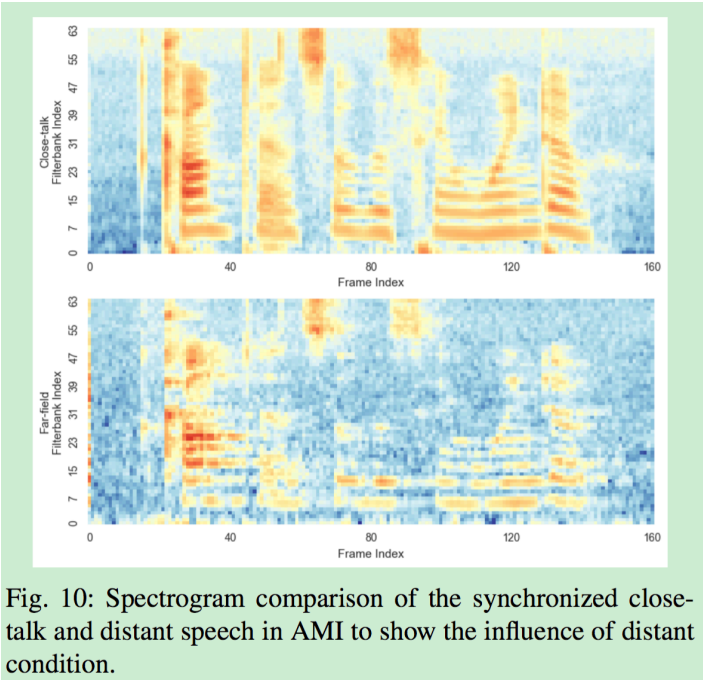
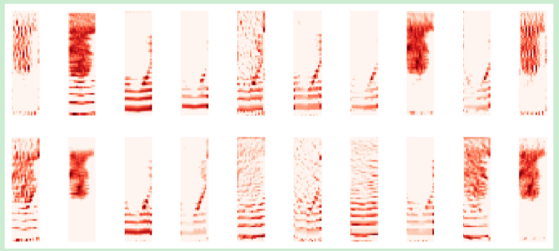


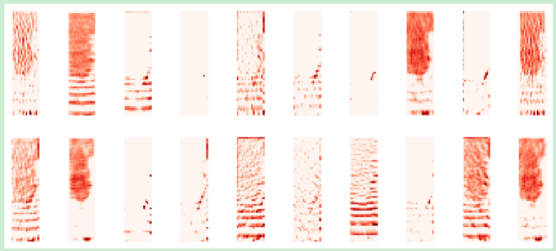
Fig. 10: Spectrogram comparison of the synchronized close-talk and distant speech in AMI to show the influence of distant condition.

Experiments

- **Evaluation on AMI (cont'd)**
- One same single synchronized frame is propagated.

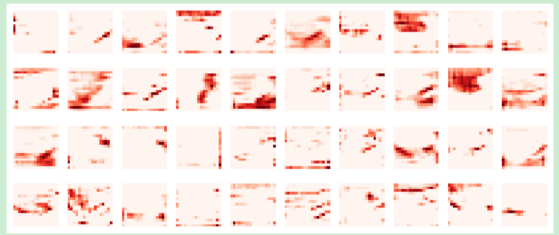


(a) Close-Talk

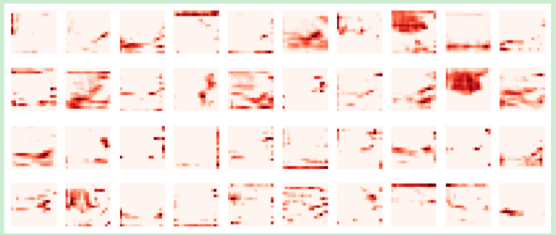


(b) Distant

Fig. 11: Selected feature maps of the first convolutional layer using the synchronized frame from close-talk and distant conditions in AMI



(a) Close-Talk



(b) Distant

Fig. 12: Selected feature maps of the sixth convolutional layer using the synchronized frame from close-talk and distant conditions in AMI

Experiments

- **Evaluation on AMI (cont'd)**

TABLE XI: WER (%) comparison of different models on AMI MDM condition

Model	dev	eval
DNN	47.5	52.3
CNN	46.3	51.3
LSTM	42.9	46.6
vd10-fpad-tpad	**42.5**	**46.9**
IBM-VGG	42.9	47.4

Content

- Abstract
- Review of Convolutional Neural Networks
- Model Description
- Experiments
- Conclusion

Conclusion

- **Features of VDCNN**
- The sizes of filters and pooling templates are small.
- The input feature maps are large.
- Other design such as pooling in time, padding, and input feature maps selection are adjusted.
- On Aurora4, it achieves a WER of 8.81% (state-of-art).
- On AMI, its accuracy is competitive to an LSTM.

Thank You!