Network	Error
Segmental NN (Abdel-Hamid et al., 2013)	21.9
MCRBM (Dahl et al., 2010)	20.5
DSR (Tang et al., 2015)	19.9
Rectifier NN (Tóth, 2013)	19.8
DBN (Srivastava et al., 2014)	19.7
Shallow CNN (Ba & Caruana, 2014)	19.5
Structured DNN (Yang et al., 2016)	18.8
Posterior Modeling (Prabhavalkar et al., 2013)	18.5
Kaldi	18.53
CS-Kaldi	18.48
IC-Kaldi	18.46
MA-Kaldi	18.51
DC-Kaldi	18.50
AC-Kaldi	18.41
CTC (Graves et al., 2013)	18.4
RNN Transducer (Graves et al., 2013)	17.7
Attention RNN (Chorowski et al., 2015)	17.6
CTC+SCRF (Lu et al., 2017)	17.4
Segmental RNN (Lu et al., 2016)	17.3
RNNDrop (Moon et al., 2015)	16.9
CNN (Tóth, 2014)	16.7

Table 2. Phone error rate (%) on the TIMIT test set.

of the art performance. We apply AC to the FNN in this recipe. The inputs of the FNN are the FMLLR (Gales, 1998) features of the neighboring 21 frames, which are mean centered and normalized to have unit variance. The number of hidden layers is 4. Each layer has 1024 hidden units. Stochastic gradient descent (SGD) is used to train the network. The learning rate is set to 0.008. We compare with four diversity-promoting regularizers: CS, IC, MA and DeCorrelation (DC) (Cogswell et al., 2015). The regularization parameter in these methods are tuned in $\{10^{-6}, 10^{-5}, \cdots, 10^{5}\}$. The β parameter in IC is set to 1.

Table 2 shows state of the art phone error rate (PER) on the TIMIT core test set. Methods in the first panel are mostly based on FNN, which perform less well than Kaldi. Methods in the third panel are all based on RNNs which in general perform better than FNN since they are able to capture the temporal structure in speech data. In the second panel, we compare AC with other diversity-promoting regularizers. Without regularization, the error is 18.53%. With AC, the error is reduced to 18.41%, which is very close to a strong RNN-based baseline Connectionist Temporal Classification (CTC) (Graves et al., 2013). Besides, AC outperforms other regularizers.

CNN for Image Classification The CIFAR-10 dataset contains 32x32 color images from 10 categories, with 50,000 images for training and 10,000 for testing. We used 5000 training images as the validation set to tune hyperparameters. The data is augmented by first zero-padding the images with 4 pixels on each side, then randomly cropping the padded images to reproduce 32x32 images. We apply AC to wide residual network (WideResNet) (Zagoruyko & Komodakis, 2016) where the depth is set to 28 and the width is set to 10. SGD is used for training, with epoch number 200, initial learning rate 0.1, minibatch size 128, Nesterov momentum 0.9, dropout probability 0.3 and weight decay 0.0005. The learning rate is dropped by 0.2 at 60, 120 and 160 epochs. The performance is the median of 5 runs. We compare with CS, IC, MA, DC and an Orthogonality-Promoting (OP) regularizer (Rodríguez et al., 2016).

Network	Error
Maxout (Goodfellow et al., 2013)	9.38
NiN (Lin et al., 2013)	8.81
DSN (Lee et al., 2015)	7.97
Highway Network (Srivastava et al., 2015)	7.60
All-CNN (Springenberg et al., 2014)	7.25
ResNet (He et al., 2016)	6.61
ELU-Network (Clevert et al., 2015)	6.55
LSUV (Mishkin & Matas, 2015)	5.84
Fract. Max-Pooling (Graham, 2014)	4.50
WideResNet (Huang et al., 2016)	3.89
CS-WideResNet	3.81
IC-WideResNet	3.85
MA-WideResNet	3.68
DC-WideResNet	3.77
OP-WideResNet	3.69
AC-WideResNet	3.63
ResNeXt (Xie et al., 2016b)	3.58
PyramidNet (Huang et al., 2016)	3.48
DenseNet (Huang et al., 2016)	3.46
PyramidSepDrop (Yamada et al., 2016)	3.31

Table 3. Classification error (%) on CIFAR-10 test set

Table 3 shows state of the art classification error on the test set. Compared with the unregularized WideResNet which achieves an error of 3.89%, applying AC reduces the error to 3.63%. AC achieves lower error than other regularizers.

LSTM for Question Answering We apply AC to long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) network, which is a type of RNN. Given the input $\mathbf{x}_t$ at timestamp t, LSTM produces a hidden state $\mathbf{h}_t$ based on the following transition equations:

$$\begin{aligned}
&\mathbf{i}_t = \sigma(\mathbf{W}^{(i)}\mathbf{x}_t + \mathbf{U}^{(i)}\mathbf{h}_{t-1} + \mathbf{b}^{(i)})\\
&\mathbf{f}_t = \sigma(\mathbf{W}^{(f)}\mathbf{x}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)})\\
&\mathbf{o}_t = \sigma(\mathbf{W}^{(o)}\mathbf{x}_t + \mathbf{U}^{(o)}\mathbf{h}_{t-1} + \mathbf{b}^{(o)})\\
&\mathbf{c}_t = \mathbf{i}_t \odot \tanh(\mathbf{W}^{(c)}\mathbf{x}_t + \mathbf{U}^{(c)}\mathbf{h}_{t-1} + \mathbf{b}^{(c)}) + \mathbf{f}_t \odot \mathbf{c}_{t-1}\\
&\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\mathbf{W}$s are $\mathbf{U}$s are gate-specific weight matrices. On the row vectors of each weight matrix, the AC is applied. The LSTM is used for a question answering (QA) task on two datasets: CNN and DailyMail (Hermann et al., 2015),