REDES CONVOLUCIONAIS DE LONGO PRAZO A CURTO PRAZO, CONEXAS NECESSÁRIAS TOTALMENTE CONECTADAS

Tara N. Sainath, Oriol Vinyals, Andrew Senior, Has¸im Sak

Google, Inc., Nova York, NY, EUA
{tsainath, vinyals e andrewsenior, hasim}@google.com

RESUMO

As redes neurais convolucionais (CNNs) e a memória de longo prazo (LSTM) mostraram melhorias em relação às redes neurais profundas (DNNs) em uma ampla variedade de tarefas de reconhecimento de fala. CNNs, LSTMs e DNNs são complementares em seus recursos de modelagem, como CNNs são bons em reduzir variações de frequência, LSTMs são bons em modelagem temporal e DNNs são apropriados para mapear recursos para um espaço mais separável. Neste artigo, aproveitamos a complementaridade de CNNs, LSTMs e DNNs combinando-os em uma arquitetura unificada. Exploramos a arquitetura proposta, que chamamos de CLDNN, em uma variedade de grandes tarefas de vocabulário, variando de 200 a 2.000 horas. Concluímos que o CLDNN fornece uma melhoria relativa de 4-6% no WER em relação a um LSTM, o mais forte dos três modelos individuais.

1. INTRODUÇÃO

Nos últimos anos, as Redes Neurais Profundas (DNNs) alcançaram um tremendo sucesso em tarefas de reconhecimento contínuo de fala de grande vocabulário (LVCSR) em comparação com os sistemas Modelo de Mistura Gaussiana / Modelo de Markov Oculto (GMM / HMM) [1]. Recentemente, foram obtidas melhorias adicionais sobre DNNs com tipos alternativos de arquiteturas de redes neurais, incluindo redes neurais convolucionais (CNNs) [2] e redes neurais recorrentes de memória de longo prazo (LSTMs) [3]. CNNs, LSTMs e DNNs são limitados individualmente em suas capacidades de modelagem, e acreditamos que o desempenho do reconhecimento de fala pode ser melhorado combinando essas redes em uma estrutura unificada.

Uma boa visão geral das limitações de modelagem de RNNs (e, portanto, LSTMs) é fornecida em [4]. Um problema com os LSTMs é que a modelagem temporária é feita no recurso de entrada x_t (isto é, recurso log-mel). No entanto, a modelagem de alto nível de x_t pode ajudar a separar fatores de variação subjacentes na entrada, o que deve facilitar a aprendizagem da estrutura temporal entre etapas sucessivas do tempo [4]. Por exemplo, foi demonstrado que as CNNs aprendem recursos adaptados ao orador / treinados discriminativamente e, assim, removem variações na entrada [5]. Portanto, pode ser benéfico proceder aos LSTMs com algumas camadas CNN totalmente conectadas.

De fato, os sistemas GMM / HMM de última geração realizam a adaptação do alto-falante, usando técnicas como normalização do comprimento do trato vocal (VTLN) e regressão linear de probabilidade máxima de espaço no recurso (fMLLR), antes de realizar a modelagem temporal via HMMs [6] Esta ordem de receita demonstrou ser apropriada para tarefas LVCSR [7]. Portanto, faz sentido explorar a passagem da entrada x_t às camadas CNN, que reduzem a variação na frequência da entrada, antes de passar para as camadas LSTM para modelar a sequência temporalmente.

Além disso, como mencionado em [4], nos LSTMs o mapeamento entre h_t e saída y_t também não é profundo, o que significa que não há intermediário camada oculta não linear. Se fatores de variação nos estados ocultos pudessem ser reduzidos, o estado oculto do modelo poderia resumir o histórico de entradas anteriores com mais eficiência. Por sua vez, isso poderia facilitar a previsão da saída. A redução da variação nos estados ocultos pode ser modelada com camadas DNN após as camadas LSTM. Isso é similar em espírito ao modelo oculto para saída proposto em [4], e também testado para fala, embora com RNNs [8].

O modelo que propomos é alimentar os recursos de entrada, cercados pelo contexto temporal, em algumas camadas da CNN para reduzir a variação espectral. A saída da camada CNN é então alimentada em algumas camadas LSTM para reduzir as variações temporais. Em seguida, a saída da última camada LSTM é alimentada para algumas camadas DNN totalmente conectadas, que transformam os recursos em um espaço que facilita a classificação da saída.

A combinação de CNN, LSTMs e DNNs foi explorada em [9]. No entanto, nesse artigo, os três modelos foram treinados primeiro separadamente e, em seguida, as três saídas foram combinadas através de uma camada de combinação. Nosso artigo é diferente, pois combinamos CNNs, LSTMs e CNNs em uma estrutura unificada que é treinada em conjunto. Além disso, nossa escolha de como combinar essas camadas é motivada pela análise em [4], o que indica que o desempenho do LSTM pode ser melhorado ao fornecer melhores recursos ao LSTM (que as camadas da CNN fornecem através da redução da variação espectral), além de melhorar as previsões de saída, aprofundando o mapeamento entre unidades e saídas ocultas (que as camadas DNN fornecem).

Cada bloco CNN, LSTM e DNN captura informações sobre a representação de entrada em diferentes escalas [10]. Portanto, exploramos se outras melhorias podem ser obtidas combinando informações em várias escalas. Especificamente, exploramos a passagem de um recurso de longo prazo para a CNN, que é passado para o LSTM junto com um recurso de curto prazo. Além disso, exploramos a complementaridade entre os recursos de modelagem das camadas LSTM e DNN. Especificamente, investigamos a passagem da saída da camada CNN para as duas camadas LSTM e DNN. Iremos nos referir à arquitetura CLDNN com essas conexões adicionais como um CLDNN multi-escala.

Nossas experiências iniciais para entender o comportamento do CLDNN são realizadas em uma tarefa de pesquisa por voz de 200 horas. Concluímos que a arquitetura CLDNN fornece uma melhoria relativa de 4% no WER em relação ao LSTM, e a inclusão de recursos em várias escalas fornece uma melhoria relativa adicional de 1%. A seguir, exploramos o comportamento da arquitetura CLDNN em duas tarefas maiores de Pesquisa por voz, ou seja, um corpus de 2.000 horas de fala limpa e um discurso de 2.000 horas de fala ruidosa. Aqui, concluímos que o CLDNN oferece uma melhoria relativa de 4-5% no WER em relação ao LSTM, e as adições em várias escalas fornecem uma melhoria relativa adicional de 1%. Isso demonstra a robustez da arquitetura CLDNN proposta com conjuntos de dados maiores e diferentes condições ambientais.

O restante deste artigo é a seguinte. Na Seção 2, descrevemos a arquitetura CLDNN, bem como as adições em várias escalas. A configuração experimental é descrita na Seção 3 e os experimentos iniciais para

entender a arquitetura CLDNN são apresentados na Seção 4. Resultados dos conjuntos de dados maiores são discutidos na Seção 5. Finalmente, a Seção 6 conclui o artigo e discute trabalhos futuros.

2. ARQUITETURA MODELO

Esta seção descreve a arquitetura CLDNN mostrada na Figura 1.

2.1 CLDNN

Na Figura 1, quadro x_t cercado por *eu* vetores contextuais à esquerda e *r* vetores contextuais à direita, é passado como entrada para a rede. Esta entrada é indicada como [$x_{t-eu}, \ldots, x_{t+r}$. Em nosso trabalho, cada quadro x_t é um recurso de banco de dados de banco de mel de 40 dimensões.

Primeiro, reduzimos a variação de frequência no sinal de entrada, passando a entrada por algumas camadas convolucionais. A arquitetura usada para cada camada da CNN é semelhante à proposta em [2]. Especificamente, usamos 2 camadas convolucionais, cada uma com 256 mapas de características. Usamos um filtro de tempo de frequência 9x9 para a primeira camada convolucional, seguido de um filtro 4x3 para a segunda camada convolucional, e esses filtros são compartilhados em todo o espaço de tempo e frequência. Nossa estratégia de agrupamento é usar o agrupamento máximo não sobreposto, e o agrupamento em frequência somente é realizado [11]. Um tamanho de pool de 3 foi usado para a primeira camada e nenhum pool foi feito na segunda camada.

A dimensão da última camada da CNN é grande, devido ao número de mapas de **características * Tempo * contexto de frequência. Assim, adicionamos uma camada linear para** reduzir a dimensão da característica, antes de passá-la para a camada LSTM, como indicado na Figura 1. Em [12], descobrimos que a adição dessa camada linear após as camadas CNN permite uma redução nos parâmetros com sem perda de precisão. Em nossos experimentos, descobrimos que reduzir a dimensionalidade, de modo a termos 256 saídas da camada linear, era apropriado.

Depois que a modelagem de frequência é realizada, passamos a saída CNN para as camadas LSTM, que são apropriadas para modelar o sinal no tempo. Seguindo a estratégia proposta em [3], usamos 2 camadas LSTM, onde cada camada LSTM possui 832 células e uma camada de projeção de 512 unidades para redução de dimensionalidade. Salvo indicação em contrário, o LSTM é desenrolado por 20 etapas de tempo para treinamento com retropropagação truncada ao longo do tempo (BPTT). Além disso, o rótulo do estado de saída está atrasado em 5 quadros, como observamos com DNNs que informações sobre quadros futuros ajudam a prever melhor o quadro atual. O **recurso de entrada na CNN possui *eu* quadros contextuais à esquerda e *r* à direita e a saída** CNN é então passada para o LSTM. Para garantir que o LSTM não veja mais de 5 quadros **de contexto futuro, o que aumentaria a latência da decodificação, definimos *r* = 0 0 para** CLDNNs.

Finalmente, após executar a modelagem de frequência e temporal, passamos a saída do LSTM para algumas camadas DNN totalmente conectadas. Como mostrado em [5], essas camadas superiores são apropriadas para produzir uma representação de recurso de ordem superior que é mais facilmente separável nas diferentes classes que queremos discriminar. Cada camada totalmente conectada possui 1.024 unidades ocultas.

2.2 Adições em várias escalas

A CNN adota um recurso de longo prazo, vendo um contexto de *t* - *eu* para *t* *(* ou seja, *r* = 0 0 no CLDNN) e produz uma representação de ordem superior para passar para o LSTM. O LSTM é então desenrolado por 20 timesteps e, portanto, consome um contexto maior de 20 + *eu*. No entanto, sentimos que há informações complementares ao passar também a curto prazo. x_t recurso para o LSTM. De fato, o trabalho original do LSTM em [3] analisou a modelagem de uma sequência de 20 sequências consecutivas de curto prazo. x_t

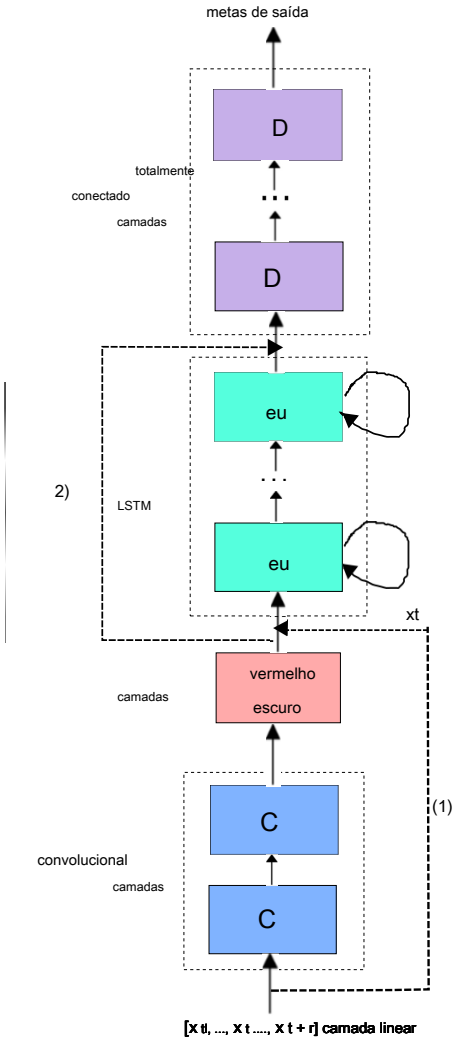

Figura 1. Arquitetura CLDNN

recursos, sem contexto. Para modelar recursos de curto e longo prazo, utilizamos o x_t e passe isso como entrada, junto com o recurso de longo prazo da CNN, para o LSTM. **Isso é mostrado pelo fluxo tracejado (1) na Figura 1.**

O uso de características de curto e longo prazo em uma rede neural já foi explorado anteriormente (isto é, [13, 14]). A principal diferença entre o trabalho anterior e o nosso é que somos capazes de fazer isso em conjunto em uma rede, principalmente devido ao poder da modelagem seqüencial LSTM. Além disso, nossa combinação de recursos de curto e longo prazo resulta em um aumento insignificante no número de parâmetros de rede.

Além disso, exploramos se há complementaridade entre modelar a saída da CNN temporalmente com um LSTM e discriminativamente com um DNN. Especificamente, motivados pelo trabalho em visão computacional [10], exploramos a passagem da saída da CNN para o LSTM e DNN. Isso é indicado pelo fluxo pontilhado (2) na Figura 1. Essa ideia de combinar informações das camadas CNN e DNN foi explorada antes na fala [11, 15], embora o trabalho anterior tenha adicionado camadas DNN extras para fazer a combinação. Nosso trabalho difere em que passamos a saída da CNN diretamente para a DNN, sem camadas extras e, portanto, um aumento mínimo dos parâmetros.

3. EXPERIÊNCIAS

Nossos experimentos iniciais para entender as arquiteturas da CNN, DNN e LSTM são realizados em um conjunto de treinamento de tamanho médio, composto por 300k de expressões em inglês (cerca de 200 horas). Outras experiências são então realizadas em um conjunto de treinamento maior de 3m de enunciados (2.000 horas). Além disso, para explorar a robustez do nosso modelo ao ruído, também realizamos experimentos usando um conjunto de treinamento barulhento de 3m de enunciados (2.000 horas). Esse conjunto de dados é criado corrompendo artificialmente as expressões limpas usando um simulador de ambiente, adicionando graus variados de ruído e reverberação, de modo que o SNR geral esteja entre 5dB e 30dB. As fontes de ruído são do YouTube e gravações ambientais barulhentas da vida cotidiana. Todos os conjuntos de treinamento são anonimizados e transcritos à mão e são representativos do tráfego de fala do Google. Os modelos treinados em fala limpa são avaliados em um conjunto de testes limpo contendo 30.000 expressões (20 horas). Além disso, os modelos treinados em fala ruidosa são avaliados em condições correspondentes em um conjunto de testes de 30.000 ruídos de expressão, ao qual o ruído em vários SNRs foi adicionado ao conjunto de testes limpo. É importante observar que os conjuntos de treinamento e teste usados neste artigo são diferentes dos de [3] e, portanto, os números não podem ser comparados diretamente.

O recurso de entrada para todos os modelos são os recursos do banco de dados do banco de dados de 40 dimensões, calculados a cada 10 ms. Salvo indicação em contrário, todas as redes neurais são treinadas com o critério de entropia cruzada, usando a estratégia de otimização de descida de gradiente estocástico assíncrono (ASGD) descrita em [16]. As experiências de treinamento em sequência neste artigo também usam ASGD distribuído, que é descrito em mais detalhes em [17]. Todas as redes possuem 13.522 destinos de saída de CD. Os pesos para todas as camadas CNN e DNN são inicializados usando a estratégia Glorot-Bengio descrita em [18]. Salvo indicação em contrário, todas as camadas LSTM são aleatoriamente inicializadas para serem gaussianas, com uma variação de

1 / (# entradas). Além disso, a taxa de aprendizado é escolhida específica para cada rede e é escolhida como o maior valor, de modo que o treinamento permaneça estável. As taxas de aprendizado são exponencialmente deterioradas.

4. RESULTADOS

Os resultados iniciais para entender a combinação do modelo CLDNN e suas variantes são apresentados nesta seção. Todos os modelos são treinados no conjunto de treinamento limpo de tamanho médio de 200 horas e os resultados são relatados no conjunto de teste limpo.

4.1 Linhas de base

Primeiro, estabelecemos números de linha de base para CNNs, DNNs e LSTMs, conforme mostrado na Tabela 1. Consistente com os resultados relatados na literatura [2], a CNN é treinada com 2 camadas convolucionais com 256 mapas de características e 4 camadas totalmente conectadas de 1.024 unidades ocultas. O DNN é treinado com 6 camadas, 1.024 unidades ocultas [1]. A entrada na CNN e na DNN é um recurso de banco de dados de banco de mel de 40 dimensões, cercado por um contexto de 20 quadros passados e 5 quadros futuros. O LSTM é treinado com 2 camadas de 832 células e uma camada de projeção de 512 dimensões. Adicionar camadas LSTM extras a essa configuração não foi encontrado para ajudar [3]. A entrada no LSTM é um recurso único de banco de dados de mel e log de 40 dimensões. O LSTM é desenrolado 20 etapas no tempo e a saída é atrasada em 5 quadros.

4.2 CNN + LSTM

Nesta seção, analisamos o efeito da adição de CNN antes do LSTM. Para mostrar o benefício das CNNs sobre os DNNs, também relatamos

Método	WER DNN
	18,4
CNN	18,0
LSTM 18.0	

Tabela 1. Linhas de base DNN, CNN, LSTM

resultados para adicionar o DNN antes da camada LSTM. A Tabela 2 compara os resultados para **CNNs e DNN, com diferentes quantidades de contexto de entrada esquerdo (ou seja, *eu)* para a** CNN e DNN. Observe que, para CNNs e DNNs, os melhores resultados são obtidos com um contexto à esquerda de 10 quadros. Um contexto maior de 20 prejudica o desempenho, provavelmente uma vez que o LSTM é desenrolado por 20 etapas de tempo, de modo que o contexto total processado pelo LSTM é 40. Observe também que os benefícios das CNNs sobre as DNNs [2] continuam mantendo-se mesmo quando combinado com LSTMs.

Contexto de entrada	# Etapas Desenrolar	WER CNN	WER DNN
l = 0, r = 0	20	17,8	18,2
l = 10, r = 0	20	17,6	18,2
l = 20, r = 0	20	17,9	18,5

Mesa 2. WER, CNN + LSTM vs. DNN + LSTM Para garantir que as melhorias no CNN + LSTM não sejam devidas a recursos contextuais extras fornecidos à CNN (e, portanto, ao LSTM), exploramos o comportamento dos LSTMs com diferentes contextos temporais. Primeiro, fornecemos o LSTM com uma entrada que abrange dez quadros à **esquerda do quadro atual (ou seja, *l* = 10), o mesmo recurso de entrada fornecido à CNN. O** LSTM ainda está desenrolado em 20 timesteps. A Tabela 3 mostra que isso não melhora o WER em relação ao fornecimento de nenhum contexto de recurso ao LSTM. Além disso, comparamos a passagem de um único quadro para o LSTM e o desenrolando por 30 etapas, mas isso degrada o WER. Isso ajuda a justificar os ganhos da arquitetura CNN + LSTM, mostrando a importância de extrair recursos mais robustos (com CNNs) antes de executar a modelagem temporal (com LSTMs).

Método	NÓS SOMOS
LSTM, *l* = 0, desenrolar = 20	18,0
LSTM, *l* = 10, desenrolar = 20	18,0
LSTM, *l* = 0, desenrolar = 30	18,2

Tabela 3. WER, modelagem temporal alternativa para LSTM

4.3 LSTM + DNN

Nesta seção, exploraremos o efeito de adicionar camadas totalmente conectadas após a saída do LSTM. Para este experimento, a entrada fornecida para cada rede é de quadro **único x_t e o LSTM é novamente desenrolado por 20 etapas de tempo. A Tabela 4 mostra** que as melhorias são obtidas, mas o desempenho parece saturar após duas camadas adicionais. Isso indica que, após a conclusão da modelagem temporal, é benéfico usar camadas DNN para transformar a saída da camada LSTM em um espaço mais discriminativo e mais fácil de prever destinos de saída.

# DNN Layers	NÓS SOMOS
0 0	18,0 (LSTM)
1 1	17,8
2	17,6
3	17,6

Quadro 4 WER, LSTM + DNN

4.4 CNN + LSTM + DNN

Nesta seção, reunimos os modelos das Seções 4.2 e 4.3, alimentando recursos em uma CNN, realizando modelagem temporal com um LSTM e, finalmente, alimentando essa saída em 2 camadas totalmente conectadas. A Tabela 5 mostra o WER para o LSTM, CNN + LSTM, LSTM + DNN e, finalmente, o modelo CLDNN combinado. A tabela indica que os ganhos da combinação das camadas CNN e DNN com o LSTM são complementares. No geral, somos capazes de obter uma melhoria relativa de 4% no WER em relação apenas ao modelo LSTM.

Método	NÓS SOMOS
LSTM	18,0
CNN + LSTM 17,6	LSTM +
DNN 17,6	
CLDNN	17,3

Quadro 5 WER, CLDNN

4.5 Melhor inicialização de peso

Mostramos que podemos obter ganhos usando as CNNs para fornecer melhores recursos antes de executar a modelagem temporal com LSTMs. Pode-se argumentar que, se os LSTMs são melhor inicializados, de modo que uma melhor modelagem temporal possa ser realizada, as CNNs são realmente necessárias? Nossos experimentos iniciais com LSTMs usam a inicialização aleatória de peso gaussiana, que produz valores próprios da rede recorrente inicial que são próximos de zero, aumentando assim as chances de gradientes de fuga [19]. Para resolver esse problema, analisamos a inicialização aleatória uniforme do **peso entre - 0,02 para 0,02 para as camadas LSTM.**

A Tabela 6 mostra que os ganhos com o modelo CLDNN ainda são mantidos, mesmo após uma melhor inicialização do peso, e o modelo CLDNN ainda possui uma melhoria relativa de 4% em relação ao LSTM. Isso justifica o benefício de ter as camadas CNN fornecendo melhores recursos para a modelagem temporal. Observe agora que, com a inicialização adequada do peso, o LSTM é melhor que o CNN ou DNN na Tabela 1.

Método WER - Gaussian Init	WER - Uniform Init
LSTM 18,0	17,7
CLDNN 17,3	17,0

Quadro 6 WER, Inicialização de Peso

4.6 Investigações em várias escalas

Nesta seção, investigamos a adição de informações em várias escalas na arquitetura CLDNN, conforme descrito na Seção 2.2. Primeiro, exploramos a aprovação de um recurso de longo **prazo [$x_{t-10}, \ldots, x_t$ CNN, e um recurso de curto prazo x_t para o LSTM. A Tabela 7 mostra que isso** fornece um WER de 16,8%, uma melhoria relativa adicional de 1% em relação à passagem apenas do recurso de longo prazo da CNN para o LSTM.

Segundo, exploramos a passagem da saída da CNN para o LSTM e o DNN. A Tabela 7 indica que isso não gera ganhos somente com o CLDNN. Isso indica que o processamento temporal dos recursos da CNN usando o LSTM é suficiente e mais informações não são obtidas ao passar adicionalmente os recursos da CNN para o DNN.

5. RESULTADOS EM MAIORES CONJUNTOS DE DADOS

Nesta seção, comparamos CLDNNs e LSTMs, bem como adições em várias escalas, em conjuntos de dados maiores. Observe quando dizemos multi-scale

Método	NÓS SOMOS
LSTM	17,7
CLDNN, recurso de longo prazo para LSTM 17.0	
+ recurso de curto prazo para LSTM	16,8
+ Camadas CNN para LSTM e DNN	17,0

Quadro 7 WER com adições em várias escalas

No CLDNN, apenas incluímos resultados passando recursos de curto e longo prazo para o LSTM e omitimos a passagem do CNN para o LSTM e DNN, pois apenas a primeira técnica mostrou ganhos na Seção 4.6. Além disso, nesta seção, relatamos números após o treinamento de entropia cruzada (EC) e sequência [17], uma estratégia que demonstrou proporcionar ganhos consistentes em relação ao treinamento em EC [20].

A Tabela 8 mostra o WER para os três modelos quando treinados em um Conjunto de dados limpos de 2.000 horas e depois avaliado em um conjunto de testes limpo. Com as adições CLDNN e multi-escala, podemos obter uma redução relativa de 6% no WER sobre o LSTM após o treinamento de CE e uma melhoria relativa de 5% após o treinamento de sequência.

Método	WER-CE	WER-Seq
LSTM	14,6	13,7
CLDNN	14,0	13.1
CLDNN multi-escala	13,8	13.1

Quadro 8 WER, Modelos Treinados em 2.000 horas, Limpar Finalmente, a Tabela 9 ilustra o WER para os 3 modelos quando treinados em um conjunto de treinamento barulhento de 2.000 horas e depois avaliados em um conjunto de testes barulhento. No nível CE, o CLDNN fornece uma redução relativa de 4% no WER em comparação com o LSTM e a inclusão de informações em várias escalas fornece uma pequena melhoria adicional. Após o treinamento em sequência, o CLDNN fornece uma melhoria relativa de 7% em relação ao LSTM. As melhorias com CLDNNs em conjuntos de dados maiores e após o treinamento em sequência demonstram a robustez e o valor do método proposto.

Método	WER-CE	WER-Seq
LSTM	20,3	18,8
CLDNN	19,4	17,4
CLDNN multi-escala	19,2	17,4

Quadro 9 WER, Modelos treinados em 2.000 horas, Barulhento

6. CONCLUSÕES

Neste artigo, apresentamos uma arquitetura combinada CNN, LSTM e DNN, que chamamos de CLDNN. A arquitetura usa CNNs para reduzir a variação espectral do recurso de entrada e depois o passa para as camadas LSTM para realizar a modelagem temporal, e finalmente o envia para as camadas DNN, que produz uma representação de recurso que é mais facilmente separável. Também incorporamos adições em várias escalas a essa arquitetura, para capturar informações em diferentes resoluções. Os resultados de uma variedade de tarefas de Pesquisa por voz do LVCSR indicam que a arquitetura CLDNN proposta fornece uma redução relativa de 4-6% no WER em comparação com um LSTM.

7. AGRADECIMENTOS

Obrigado a Izhak Shafran pela ajuda nos scripts de treinamento do LSTM, a Hank Liao pela ajuda nas configurações de decodificação e a Arun Narayanan pelas sugestões sobre como treinar e decodificar em condições ruidosas.

8. REFERÊNCIAS

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, TN Sainath e B. Kingsbury, "Redes neurais profundas para modelagem acústica no reconhecimento de fala" *Revista de Processamento de Sinais IEEE,* vol. 29, n. 6, pp. 82–97, 2012.

[2] TN Sainath, A. Mohamed, B. Kingsbury e B. Ramabhad- "Redes Neurais Convolucionais Profundas para LVCSR", em *Proc. ICASSP,* 2013.

[3] H. Sak, A. Senior e F. Beaufays, "Memórias de Longo Prazo arquiteturas de redes neurais recorrentes para modelagem acústica em larga escala " *Proc. Interspeech,* 2014.

[4] R. Pascanu, C. Gulcehre, K. Cho e Y. Bengio, "Como Construa redes neurais recorrentes profundas ", em *Proc. ICLR,* 2014.

[5] A. Mohamed, G. Hinton e G. Penn, "Compreensão como as redes de crenças profundas realizam modelagem acústica ", em *ICASSP,* 2012.

[6] H. Soltau, G. Saon e B. Kingsbury, "O discurso da IBMAttila kit de ferramentas de reconhecimento " *Proc. Workshop do IEEE sobre tecnologia de idiomas falados,* 2010.

[7] TN Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo e D. Kanevsky, "Recursos de representação esparsa baseados em exemplos: do TIMIT ao LVCSR" *no TSEE do IEEE,* 2011.

[8] G. Saon, H. Soltau, A. Emami e M. Picheny, "Unfolded Redes neurais recorrentes para reconhecimento de fala ", em *Inter- fala,* 2014.

[9] L. Deng e J. Platt, "Ensemble Deep Learning for Speech Reconhecimento ", em *Proc. Interspeech,* 2014.

[10] P. Sermanet e Y. LeCun, "Tráfego de reconhecimento de sinais com redes convolucionais em várias escalas " *Proc. IJCNN,* 2011.

[11] TN Sainath, B. Kingsbury, A. Mohamed, G. Dahl, G. Saon, H. Soltau, T. Beran, A. Aravkin e B. Ramabhadran, "Melhorias nas redes neurais convolucionais profundas para LVCSR", em *em Proc. ASRU,* 2013.

[12] TN Sainath, V. Peddinti, B. Kingsbury, P. Fousek, D. Na- hamoo e B. Ramabhadhran, "Espectros de dispersão profunda com redes neurais profundas para tarefas LVCSR", em *Proc. Inter- fala,* 2014.

[13] P. Schwarz, P. Matejka e J. Cernocky, "Hierarchical Struc- redes neurais para reconhecimento de fonemas ", em *Proc. ICASSP,* 2006.

[14] F. Grezl e M. Karafat, "Semi-Supervisioned Bootstrapping Ap- abordagem do treinamento para extrator de recursos de rede neural ", em *Proc. ASRU,* 2013.

[15] H. Soltau, G. Saon e TN Sainath, "Treinamento conjunto de Redes Neurais Convolucionais e Não Convolucionais ", em *em Proc. ICASSP,* 2014.

[16] J. Dean, GS Corrado, R. Monga, K. Chen, M. Devin, QV Le, MZ Mao, M. Ranzato, A. Sênior, P. Tucker, K. Yang e AY Ng, "Redes profundas distribuídas em larga escala", em *Proc. NIPS,* 2012.

[17] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior e M. Bacchiani, "Otimização estocástica assíncrona para treinamento de seqüência de redes neurais profundas", em *Proc. ICASSP,* 2014.

[18] X. Glorot e Y. Bengio, "Entendendo a dificuldade de Treinamento de redes neurais profundas de avanço ", em *Proc. AIS-TATS,* 2014.

[19] R. Pascanu, T. Mikolov e Y. Bengio, "Sobre a dificuldade de Treinamento de redes neurais recorrentes " *Proc. ICML,* 2013.

[20] B. Kingsbury, "Otimização baseada em treliça de classes de sequência Critérios de certificação para modelagem acústica de redes neurais ", em *Proc. ICASSP,* 2009.