Relatório Deep Learning
Assignment 3: Action Recognition

Ana Carolina Erthal & Felipe Lamarca

Link para o Google Colab

1 Modificações no código

Iniciamos passando pelo código já disponível no notebook para carregamento e pré-processamento dos dados, atribuindo os *labels* corretos às imagens e realizando a divisão em conjunto de treino e de teste. Em seguida, definimos o *encoder*, uma VGG16 pré-treinada na Imagenet, mantendo todas as camadas *freezed*, e adicionando uma camada de pooling (Global Average Pooling) para obtenção dos *feature vectors*. Essa etapa funciona como *feature extractor* do modelo, e formamos nosso conjunto de treino e teste.

Em seguida, elaboramos a estrutura da LSTM (Long Short Term Memory), variando o número de *units* conforme determinado para testes, adicionando ao final uma camada densa utilizando a função de ativação softmax, para classificação. Determinamos, ainda, a função de perda e o otimizador de acordo com o determinado pelo assignment, e estabelecemos um *callback* de *early stopping*.

Definimos também as estruturas do modelo utilizando uma GRU (Gated Recurrent Unit) ao invés da LSTM, adicionando novamente uma camada densa para classificação e mantendo todos os hiperparâmetros, e também utilizando uma LSTM densa, isto é, usando duas camadas de LSTM *stacked*, sendo a primeira "N-to-N".

Para cada RNN, treinamos um modelo com diferentes números de units calculando tempo de treinamento e verificando o número de epochs necessárias. Computamos as métricas (F1 score por classe, F1 score médio e acurácia) em relação ao conjunto de teste. Discutiremos mais adiante os hiperparâmetros utilizados e resultados obtidos. Ao analisarmos as métricas e definirmos o melhor modelo, alteramos o código adicionando esse modelo para classificar os vídeos.

2 Hiperparâmetros

Para todos os resultados apresentados nas tabelas de resultados abaixo, utilizamos os seguintes parâmetros:

Hiperparâmetro	**Valor**
`internal units LSTM`	50, 100, 200 e 500
`batch size`	50
`min_delta`	0.0001
`early stopping patience`	15
`# epochs`	1000
`loss`	Categorical Cross Entropy
`optimizer`	Adam
`learning rate`	0.001

Tabela 1 – Hiperparâmetros

3 Resultados

Métricas calculadas a partir de cada modelo e quantidade de *units*:

RNN	**Acc.**	**Avg. F1 Score**	**train_time (sec)**	**# params**	**# epochs**
LSTM (50)	91.97%	91.82%	6.08	112,804	30
LSTM (100)	90.47%	90.29%	5.50	245,604	18
LSTM (200)	94.98%	95.00%	7.21	571,204	18
LSTM (500)	93.98%	93.88%	9.18	2,028,004	17
GRU (50)	91.72%	91.56%	3.67	84,804	22
GRU (100)	93.73%	93.67%	4.65	184,604	18
GRU (200)	91.47%	91.36%	3.50	429,204	17
GRU (500)	91.72%	91.59%	5.83	1,523,004	17
Deep LSTM (50)	92.23%	92.05%	5.90	133,004	26
Deep LSTM (100)	87.96%	87.36%	6.73	326,004	24
Deep LSTM (200)	92.98%	92.87%	6.74	892,004	19
Deep LSTM (500)	93.65%	93.65%	5.87	4,030,004	16

Tabela 2 – Especificações e métricas de cada modelo

A tabela abaixo apresenta, para cada modelo, o F1 Score obtido para cada classe:

RNN	**Class 1** Basketball	**Class 2** Diving	**Class 3** GolfSwing	**Class 4** Skiing
LSTM (50)	84.32%	94.06%	93.71%	95.18%
LSTM (100)	83.51%	92.59%	91.86%	93.19%
LSTM (200)	90.05%	94.00%	97.00%	98.94%
LSTM (500)	88.04%	94.06%	96.00%	97.43%
GRU (50)	84.61%	91.07%	95.56%	95.00%
GRU (100)	88.04%	92.66%	96.03%	97.93%
GRU (200)	81.72%	91.81%	95.09%	96.80%
GRU (500)	84.78%	92.01%	94.17%	95.38%
Deep LSTM (50)	84.49%	95.41%	91.91%	96.41%
Deep LSTM (100)	75.00%	94.33%	86.51%	93.59%
Deep LSTM (200)	86.77%	94.83%	94.89%	95.00%
Deep LSTM (500)	88.29%	94.49%	96.03%	95.78%

Tabela 3 – F1 Score de cada classe, em cada modelo

A LSTM é uma rede desenvolvida com o objetivo de atacar o problema do vanishing gradient e da consequente memória de curto prazo, e sua arquitetura mais complexa exerce o papel de estabelecer dependências de longo prazo - fator importante para nossos dados, já que a relação entre frames distantes pode ser determinante para a classificação. O modelo obteve resultados muito bons na classificação das classes a partir dos frames, demonstrando a relevância da inovação desse modelo utilizando *cell states*.

A GRU, por sua vez, têm o mesmo objetivo, mas de forma simplificada, reduzindo complexidade de implementação e, em teoria, tempo de treinamento. Observe que, de fato, obtivemos resultados muito satisfatórios utilizando essa RNN quando comparado à LSTM padrão (piora quase indiferente), e justamente por sua simplificação de arquitetura, realizamos o treinamento muito mais rápido, e contamos com menos parâmetros.

Por fim, utilizar uma arquitetura definindo uma Deep LSTM aumenta essa complexidade, mas possibilita o aprendizado de padrões ainda mais complexos nos dados sequenciais. Observe que, de fato houve um aumento significativo no número de parâmetros dessa rede, principalmente quando há muitas *units*.

O F1-Score, combinação equilibrada entre precisão e recall, pode ser observado classe a classe na tabela 3 e sua média por modelo na tabela 2. Em termos de desempenho para essa métrica, observa-se que os desempenhos em geral foram bastante bons, e o F1 score médio ficou bastante próximo para todos os modelos e *unit values*.

Entre eles, destaca-se a LSTM padrão com 200 units, que obteve também a melhor acurácia, seguida pela sua versão com 500 units. Esse desempenho possivelmente se justifica pela apresentação dos dados: ainda que haja dependências de longo prazo, provavelmente esse padrões não eram complexos o suficiente para que houvesse necessidade de uma Deep LSTM, ou tão simples que a GRU fosse o ideal. No entanto, as métricas foram muito próximas, e realizar diagnósticos definitivos desse tipo dependeria de performances

mais distintas.

Figura 1 – Classificação do modelo LSTM (200)

Em geral, quando observamos o incremento na quantidade de *units* dentro de cada modelo, há uma melhora (ainda que leve). No LSTM, vamos de acurácia 91.97% (50 *units*) para 93.98% (500 *units*), e algo parecido ocorre na Deep LSTM, mas também aumentamos significativamente o tempo de treinamento. De fato, ao aumentarmos a quantidade de *units*, aumentamos a capacidade do modelo criar possíveis relações implícitas entre frames, e é possível que novas relações melhorem a acurácia. No modelo GRU, no entanto, não observamos melhora considerável conforme aumentamos o número de *units* (possivelmente por sua maior simplicidade), e tivemos apenas aumento de tempo de treinamento.

Por fim, a partir da tabela de F1 Score classe a classe (tabela 3), pode-se observar que o desempenho foi um pouco pior para a classe 1 (basquete), e bastante parecido para as outras três classes. Pela ocorrência dessa piora para classe 1 em todos os modelos, inferimos que provavelmente nos deparamos com uma dificuldade estabelecida pelos próprios dados, em que o movimento realizado no basquete não é tão diferenciável de outra classe.