UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JOSÉ ÍCARO SANTANA BERNARDES

AVALIAÇÃO DE GRAVIDADE DE PACIENTES DE COVID-19 COM O USO DE DEEP LEARNING

SOBRAL

2021

JOSÉ ÍCARO SANTANA BERNARDES

AVALIAÇÃO DE GRAVIDADE DE PACIENTES DE COVID-19 COM O USO DE DEEP LEARNING

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior

SOBRAL

2021

À minha família, por sua capacidade de acreditar em mim e investir em mim. Aos meus colegas e professores que me ajudaram e fizeram parte da minha jornada.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

A pandemia de covid-19 tem gerado inúmeras perdas a nossa a sociedade. Desde do seu começo, a comunidade cientifica global tem se esforçado para interromper e conter o danos gerados pela doença. Uma questão muito importante para o tratamento é avaliar de maneira rápida e segura o estado do paciente com o intuito de direcionar o melhor tratamento e auxiliar na gestão de recursos hospitalares. Atualmente, existem diversas abordagens com o uso de técnicas de Depp Learning em exames de imagem como Tomografias Computadorizadas para trazer soluções para essa questão. Este trabalho busca explorar algoritmos poucos comuns como ConvLSTM e CNN-3D, visando trazer conhecimentos novos a respeito da problemática. Os resultados alcançados mostram que o arquitetura de ConvLSTM atingiu resultado melhores que a de CNN-3D. Entretanto, se comparado a resultados de outros artigos, os algoritmos em questão ficaram abaixo.

Palavras-chave: Covid-19. Deep Learning. Tomografias Computadorizadas. ConvLSTM. CNN-3D

ABSTRACT

The covid-19 pandemic has countless losses in our society. From the beginning, scientific globally has to become a disease to stop and contain the damages of its generated. A very important issue for treatment is to quickly and safely assess the patient's condition in order to direct the best treatment and assist in the management of hospital resources. Currently, there are several approaches using Deep Learning techniques in imaging exams such as Computed Tomography(CT) to bring solutions to this issue. This work seeks to explore a few common ones such as ConvLSTM and CNN-3D, bringing new knowledge about the problem. The results show that the ConLSTM architecture performs better than CNN-3D. However, when compared to the results of other articles, the risks in question were below.
Keywords: Covid-19. Deep Learning. Computed Tomography. ConvLSTM. CNN-3D

LISTA DE ABREVIATURAS E SIGLAS

OMS	Organização Mundial de Saùde
IA	Inteligência Artificial
ML	Machine Learning
DL	Deep Learning
TC	Tomografia Computadorizada
CNN	Convolution Neural Network
LSTM	Long Short-Term Memory
COR	Características Operacionais do Receptor
AAC	Área Abaixo da Curva

SUMÁRIO

1 INTRODUÇÃO

O dia 1º de dezembro de 2019 foi considerado a data do primeiro caso de covid-19 do mundo. Ele aconteceu na província de Wuhan, na China. A doença se espalhou rápido, levando a Organização Mundial de Saùde (OMS), no dia 11 de março de 2020, a declarar que se tratava de uma pandemia. Com isso, houve uma mobilização da comunidade científica mundial de diversas áreas para encontrar soluções para o problema.

Muitos pesquisadores, não só da área da saúde, tem explorado o campo da Inteligência Artificial (IA) como ferramenta para solucionar problemas como: Diagnóstico e o Prognósticos da doença. Uma das formas de estabelecer prognósticos é saber a situação atual do paciente, ou seja, avaliar se o quadro do doente é um caso leve ou grave. Este tipo de informação ajuda os médicos ou equipe médica a tomar a melhor decisão quanto ao tratamento.

Conforme observado em (MOHAMMADI YINGXU WANG, 2021), as principais técnicas para fazer avaliação de severidade consiste na aplicação de *Machine Learning (ML)*. Nesse quesito, os algoritmos de *Deep Learning (DL)* são as principais ferramentas de estudo visto os seus resultados para análise de dados médicos obtidos nos últimos anos.

Existem dois métodos principais para se fazer a avaliação de gravidade: classificação e quantificação. A classificação consiste em analisar dados e identificar os casos graves e não graves de um paciente (MOHAMMADI YINGXU WANG, 2021). O método de quantificação consiste em calcular os danos causados ao pulmão e determinar a severidade da doença (MOHAMMADI YINGXU WANG, 2021).

O objetivo desse trabalho é avaliar se com o uso de Tomografia Computadorizada (TC) e técnicas de DL diferentes é possível chegar a resultados que agreguem conhecimento no ponto de vista de determinar a gravidade dos casos de covid-19. Com isso, contribuir com uma ferramenta de suporte a médicos e hospitais e no gerenciamento da pandemia.

1.1 Justificativas

Com o avanço de estudos estatísticos e epidemiológicos no campo da medicina foi possível estabelecer quadros da doença de maneira mais precisa e confiável. Esse tipo de informação ajuda o médico e/ou o corpo médico que trata do paciente a tomar melhores decisões quanto ao tratamento que estão aplicando e também nas questões gerenciais da pandemia.

No caso da covid-19, atualmente existem diversos trabalhos com o intuito de auto-

matizar o processo de avaliação do paciente. Em (FENG *et al.*, 2021), vemos o desenvolvimento de um *framework* que alcança resultados significativos em identificar os casos como: grave e não grave, usando TC.

Existem outras abordagens, ainda explorando as TC, como em (LI ZHENG ZHONG, 2020) que usa técnicas de segmentação com o objetivo de extrair características das imagens antes de combiná-las para gerar a classificação em caso grave e não grave.

Este trabalho propõem a exploração de algoritmos de DL em TC para avaliar a gravidade do paciente que são pouco explorados. Com isso, trazer novos conhecimentos para área de estudo e gerar novos métodos de analise para médicos e gestores e assim, ampliar o conhecimento na área.

1.2 Objetivo Geral

Explorar o uso de técnicas poucos usadas na tentativa de estabelecer à avaliação da gravidade do caso de covid-19 de um paciente.

1.3 Objetivos Específicos

- Explorar os algoritmos de ConvLSTM e *Convolution Neural Network (CNN)*-3D em busca de um modelo que melhor resolvesse o problema.
- Avaliar o uso de regularização para otimizar a ConvLSTM e CNN-3D.
- Comparar com os resultados existentes na literatura e verificar se os algoritmos alcançam resultados satisfatório.

2 MATERIAIS E MÉTODOS

2.1 Base de Dados

A base de dados utilizada consiste em um grupo de TC obtidos de 2 hospitais chineses, o Union Hospital e o Hospital Liyuan (LEI, 2020). Ele é mais conhecido como iCTCF. Esse conjunto nos fornece um grupo de imagens de pacientes atendidos pelos 2 hospitais. Ele possui tomografias separadas por gravidade: leve, regular, severo, crítico e desconhecido.

2.1.1 Análise dos dados

Os dados possuem um total de 757 tomografias. A Tabela 1 traz a quantidade de amostra disponível para cada tipo de gravidade.

Tabela 1 – Tabela com a quantidade de amostras para cada tipo

Tipo	Quantidade
Leve	6
Regular	317
Severo	141
critico	21
Desconhecido	272

Outro ponto a ser avaliado é a mortalidade para cada tipo, ou seja, a averiguar quantidade de mortos em cada caso. Isso pode mostrar algum nível de semelhança entre os tipos. Com a Tabela 2 vemos que o tipo severo possui uma taxa de mortalidade mais próxima dos casos regulares e leves. Esse comportamento pode indicar que existe uma semelhança maior entre as tomografias dos casos severos com regulares e leves do que a com os casos mais críticos.

Tabela 2 – Tabela com a quantidade de amostras para cada tipo

Tipo	Cura	Óbito
Leve	100%	0%
Regular	100%	0%
Severo	96,45%	3,55%
Crítico	38,1%	61,9%

2.2 CNN

A arquitetura de uma CNN é formada pelo empilhamento de algumas camadas convolucionais intercalada ou não, com camadas de *pooling* e no final uma camada totalmente conectada (KHOZEIMEH DANIAL SHARIFRAZI, 2021). A imagem vai ficando menor e mais profunda a medida que avança na rede. Quando chega ao final, obtemos uma previsão, no caso de um problema de classificação, temos a probabilidade de pertencimento a cada classe. A Figura 1 mostra esse processo.

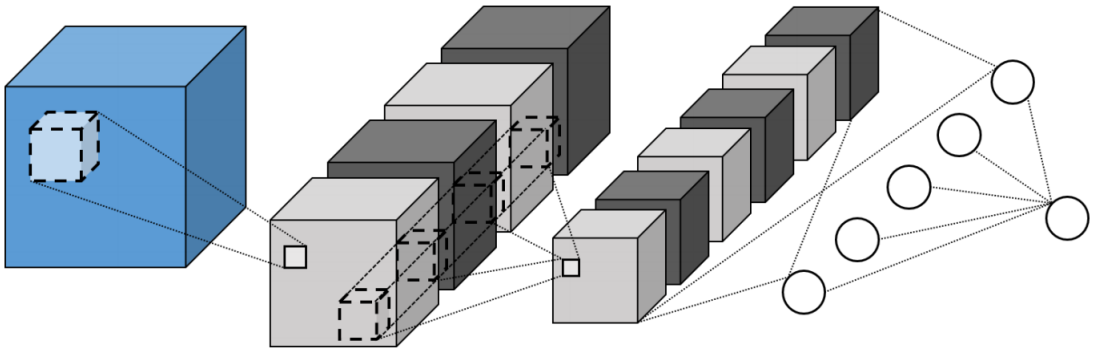


Figura 1 – Exemplo de CNN para o caso tri-dimensional (TAN; LIU, 2021). Fonte:(ALAKWAA, 2018)

A camada convolucional é composta pelo associação de filtros. Estes, que se conectam, são interpretados como uma matriz de pesos que percorrem a matriz dos dados aplicando a operação de convolução(KHOZEIMEH DANIAL SHARIFRAZI, 2021). A camada de *pooling* consiste em sub amostrar os dados a partir de alguma métrica. Uma das métricas mais comuns é a de Máximo, ou seja, percorremos os dados e coletamos o maior valor dentro do grupo de informações presentes. A camada totalmente conectada funciona como Perceptron multicamadas tradicional (KHOZEIMEH DANIAL SHARIFRAZI, 2021).

2.3 ConvLSTM

A ConvLSTM é uma variação da *Long Short-Term Memory (LSTM)* tradicional. Primeiramente, a LSTM tradicional tem as informações dispostas em 2 estados que são: estado de curto prazo (h_t) e o de longo prazo (c_t)(SAK ANDREW SENIOR, 2014). O funcionamento dessa rede consiste em fornecer a entrada atual x_t e h_t como entrada em 4 camadas: camada principal, porta de esquecimento, porta de entrada e porta de saída. A ideia é que a rede aprenda

as informações que devem ser colocadas no estado de longo prazo, o que deve ser descartado e o que ler a partir dele (SAK ANDREW SENIOR, 2014). A Figura 2 traz a maneira como a LSTM atua.

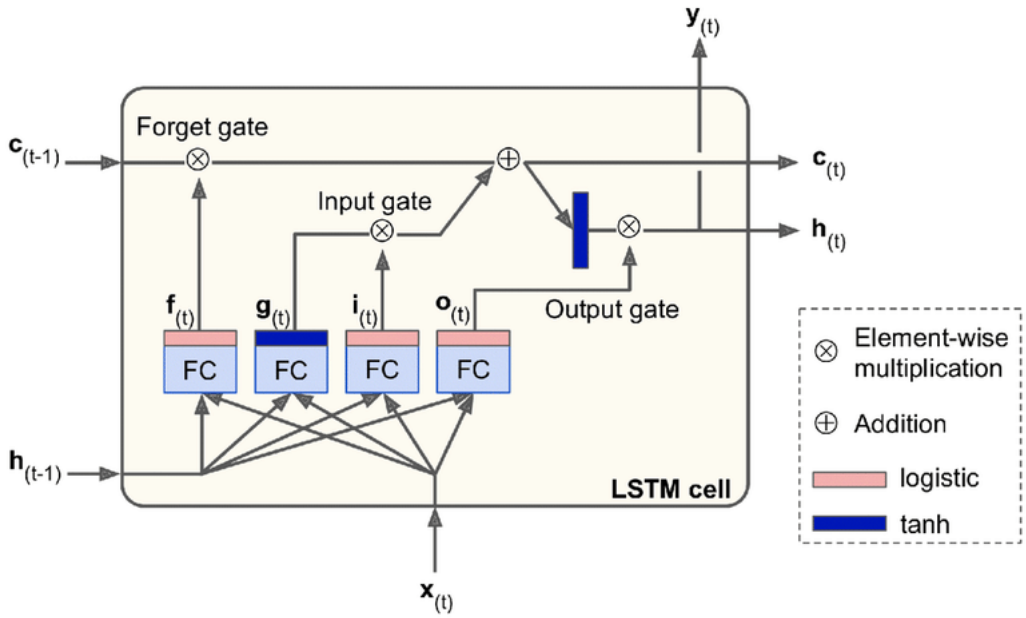


Figura 2 – LSTM. Fonte: adaptado de (GERON, 2019)

A camada principal (g_t) atua analisando a entrada atual (x_t) e o estado de curto prazo anterior (h_{t-1}), e o resultado segue para a saída da camada e no estado de curto prazo atual(h_t)(DASTIDER FARHAN SADIK, 2021). A porta de esquecimento (f_t) consiste em determinar o que devem ser eliminado do estado de longo prazo (c_t) (DASTIDER FARHAN SADIK, 2021). A porta de entrada (i_t) determina as partes de g_t que são colocadas a c_t. A porta de saída (o_t) consiste em determinar quais partes devem ser lidas e exibidas nesta celula LSTM. Da equação 2.1 a 2.6 podemos observar como todas essas variáveis são calculadas.

$$i_t = \sigma(W_{xi}^T.x_t + W_{ht}^T.h_{t-1} + b_i) \tag{2.1}$$

$$f_t = \sigma(W_{xf}^T.x_t + W_{hf}^T.h_{t-1} + b_f) \tag{2.2}$$

$$o_t = \sigma(W_{xo}^T.x_t + W_{ho}^T.h_{t-1} + b_o) \tag{2.3}$$

$$g_t = tanh(W_{xg}^T.x_t + W_{hg}^T.h_{t-1} + b_g) \tag{2.4}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{2.5}$$

$$y_t = h_t = o_t \otimes tanh(c_t) \tag{2.6}$$

Onde W^t representa a matriz transposta de pesos, b_x é o viés adotado em cada equação e o σ é a saída da função de ativação.

A ConvLSTM segue a mesma ideia da LSTM tradicional (SHI ZHOURONG CHEN, 2015). A única diferença se encontra nas equações. Ao invés de multiplicação, como da equação 2.1 a equação 2.4, temos convoluções como mostra da equação2.7 a 2.10. As equações 2.11 e 2.12 continuam iguais as 2.5 e 2.6, respectivamente.

$$i_t = \sigma(W_{xi}^T * x_t + W_{ht}^T * h_{t-1} + b_i) \tag{2.7}$$

$$f_t = \sigma(W_{xf}^T * x_t + W_{hf}^T * h_{t-1} + b_f) \tag{2.8}$$

$$o_t = \sigma(W_{xo}^T * x_t + W_{ho}^T * h_{t-1} + b_o) \tag{2.9}$$

$$g_t = tanh(W_{xg}^T * x_t + W_{hg}^T * h_{t-1} + b_g) \tag{2.10}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{2.11}$$

$$y_t = h_t = o_t \otimes tanh(c_t) \tag{2.12}$$

2.4 Medidas de Desempenho

2.4.1 Matriz de Confusão

Uma das formas de medir o desempenho de um classificador é observando sua matriz de confusão. Ela consiste em uma matriz em que as linhas correspondem a classe real enquanto que as colunas representam a classe prevista pelo algoritmo (GERON, 2019). Com isso, temos informações importantes sobre como as amostras foram classificadas corretamente(verdadeiros positivos e verdadeiros negativos) e as que não foram (falsos positivos e falsos negativos). A partir disso, podemos calcular outras métricas importantes como a acurácia, a precisão e a revocação. A acurácia consiste em quantas amostras foram classificadas de maneira correta. A equação 2.13 apresenta a maneira como se calcula em que VP representa o grupo dos verdadeiros positivos, VN verdadeiro negativos, FP falso positivo e FN falso negativo.

$$acurácia = \frac{VP+VN}{VP+VN+FP+FN} \tag{2.13}$$

A precisão consiste na acurácia das previsões positivas (GERON, 2019). Ela é dada pela equação 2.14.

$$precisão = \frac{VP}{VP+FP} \tag{2.14}$$

A revocação consiste na taxa de amostras positivas detectadas corretamente pelo classificador. Outro nome que damos para a revocação é sensibilidade. Ela é dada pela equação 2.15.

$$revocação = \frac{VP}{VP+FN} \tag{2.15}$$

2.4.2 COR

Outra métrica importante para a análise de um classificador, é a curva Características Operacionais do Receptor (COR). Ela consiste em um gráfico que plota a sensibilidade versus 1 - especificidade(GERON, 2019). A especificidade é a razão das instancias negativas classificadas corretamente como negativas (GERON, 2019).

Para compararmos os classificadores, basta calcularmos à Área Abaixo da Curva (AAC). Se o AAC do algoritmo for igual a 1, ele será perfeito. Enquanto que um AAC igual 0.5, ele será puramente aleatório.

2.4.3 Dropout

O *dropout* é a técnica de regularização mais comum para DL (GERON, 2019). Aplicar regularização ajuda a evitar problemas de sobreajuste ao conjunto de dados. O sobreajuste é um termo estatístico para descrever comportamento do modelo quando o mesmo se ajusta ao conjunto de dados apresentados anteriormente e o modelo não tem uma capacidade de prever valores novos.

O *dropout* consiste em descartar uma certa porcentagem dos nerônios em cada etapa de treinamento(DABBURA, 2018). A escolha dos neurônios que serão descartados é feita de maneira aleatória e cada neurônio tem a mesma probabilidade de ser descartado(DABBURA, 2018). Os neurônios descartados podem voltar ao processo de treinamento na etapa seguinte. A Figura 3 traz uma visão geral. Nela observamos os neurônios marcados em vermelho que estão de fora dessa etapa de treinamento.

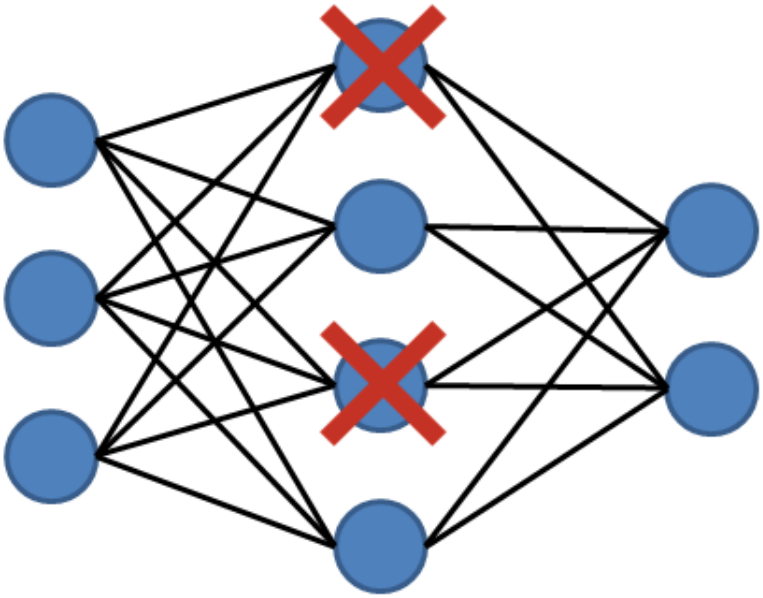


Figura 3 – *Dropout*. Fonte: adaptado de (DABBURA, 2018).

3 METODOLOGIA

O fluxograma trazido na Figura 4 mostra as etapas de desenvolvimento do projeto.

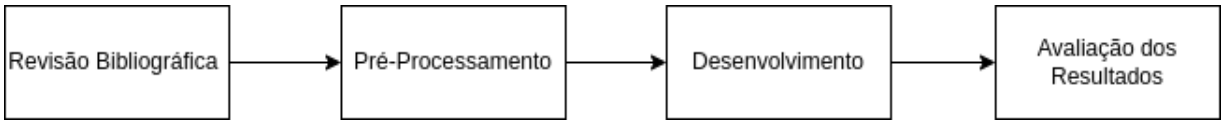


Figura 4 – Fluxograma do projeto. Fonte: elaborada pelo autor.

Na revisão bibliografia é feito um estudo das técnicas mais utilizadas e resultados já encontrados, visando fazer uma levantamento das técnicas já utilizadas e mostrar como o trabalho se encaixa no cenário atual e a sua contribuição a cerca do tema estudado.

Pela base de dados vemos que não é possível testar o modelo sem fazer ajuste nos exames. O primeiro problema é das dimensões das fatias que foram reduzidas de 512 x 512 para 64 x 64. Isso foi necessário para poder treinar os modelos. O principal problema está na quantidade de fatias presentes em cada tomografia. A média de quantidade de fatias nas tomografias é de 277. Para fazer essa padronização foram consideradas duas possibilidades: deixar todas as fatias com numero de fatias equivalente ao mínimo ou escolher um número de fatias e descartar as tomografia que possuem menos.

Para decidir, priorizou-se a técnica que gerasse a menor perda de informação por imagem. A métrica para dimensionar perda de informação por imagem foi o volume da tomografia. O volume foi tratado como o prisma de altura x largura x fatias. O volume de referência é de um prisma de dimensões 64 x 64 x 277. A técnica de reduzir para o mínimo gera um volume de equivalente a 18% do referência. O número escolhido para as fatias é 150, pois resulta na menor perda de imagens. Com essa quantidade de fatias temos um volume equivalente de 54% do volume de referencia.

No processo de Desenvolvimento treinamos os algoritmos de ConvLSTM e CNN-3D. Para isso, o conjunto de dados foi divido em 2 grupos: treino e teste. O grupo de treino ficou com 90% das imagens disponíveis e 10% para teste. O conjunto de treino foi ainda retirado 12% para validação. Depois foram rotulados em 2 classes: leve/regular e severo/crítico.

Na última etapa, temos a avaliação de resultados. Nesta fase, os resultados coletados são avaliados com o intuitos de descobrir os melhores. Depois esses resultados foram comparados com os encontrados na literatura para verificar se os resultados foram satisfatórios. O resultado sera considerado satisfatório se tiver uma diferença de $\pm$3% nas métricas de avaliação em comum.

4 RESULTADOS

O desenvolvimento foi feito com a linguagem *python* 3.7 e com o *TensorFlow* 2.7. O ambiente de desenvolvimento foi o *Google Colab Pro*. Os modelos escolhidos para fazer a exploração dos dados foram: ConvLSTM e CNN-3D. Eles compartilham os mesmos tamanhos de lote e épocas. A partir dos testes realizados, pegamos o modelo que apresentou o melhor desempenho para cada algoritmo testado e comparamos com os resultados encontrados na literatura .

A acurácia do ConvLSTM foi de 81%, enquanto que a CNN-3D obteve 73%. No caso da precisão, a ConvLSTM obteve 89% o que mostra uma confiança de que se a rede identifica o caso como Severo/Crítico, a probabilidade de ser realmente é bem alta. A CNN-3D alcança uma precisão de 81%. Na Revocação, a ConvLSTM atinge 76% e a CNN-3D obtém 65%, o que mostra a dificuldade dos algoritmos em discernir os casos Severo/Críticos dos casos Regular/Leve.

Uma consequência da precisão e da revocação é que a média harmônica (F1-score) da ConvLSTM foi de 82% e a da CNN-3D tem 76%. No que se trata da AAC, a ConvLSTM obtém 82% e a CNN-3D 74% , o que faz com que a ConvLSTM é o modelo que mais se aproxima do classificador perfeito.

A Tabela 3 traz um resumo dos resultados e uma comparação com os resultados encontrados na literatura.

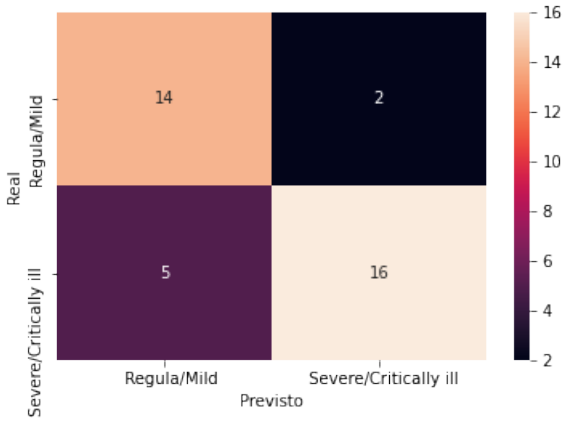
Tabela 3 – Tabela com os melhores resultados.

Modelo	Acurácia	Precisão	Revocação	F1-score	AAC
ConvLSTM	81%	89%	76%	82%	82%
CNN	73%	81%	65%	72%	74%
(FENG *et al.*, 2021)	94%	-	82%	-	90%
(LI ZHENG ZHONG, 2020)	-	-	94%	-	97%

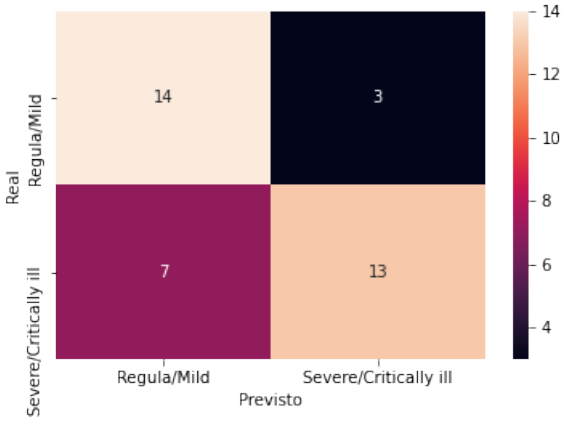
Conforme vemos na Tabela 3, dentre os algoritmos testados vemos que o ConvLSTM alcançou os melhores resultados. Porém, os resultados alcançados ainda estão longe do ideal se comparado com os de outros trabalhos.

Outro ponto para compreender melhor o funcionamento dos modelos, é a sua matriz de confusão trazido na Figura 5. Nela notamos que tanto o modelo de ConvLSTM como o de CNN-3D compartilham do mesmo problema. Ambas as redes classificam de maneira errada mais casos do tipo Severo/Crítico como Regular/leve do que casos.

Conforme vemos na Figura 5, o modelo de ConvLSTM classifica 5 amostras como Regular/leve que correspondem a Severo/Crítico na realidade. Na CNN-3D, o modelo classifica 7 amostras do tipo Severo/Crítico como Regular/Leve. Isso é mais um indício de que os casos severos tem uma semelhança maior com os regulares e leves do que com os casos mais críticos da doença.



(a) ConvLSTM



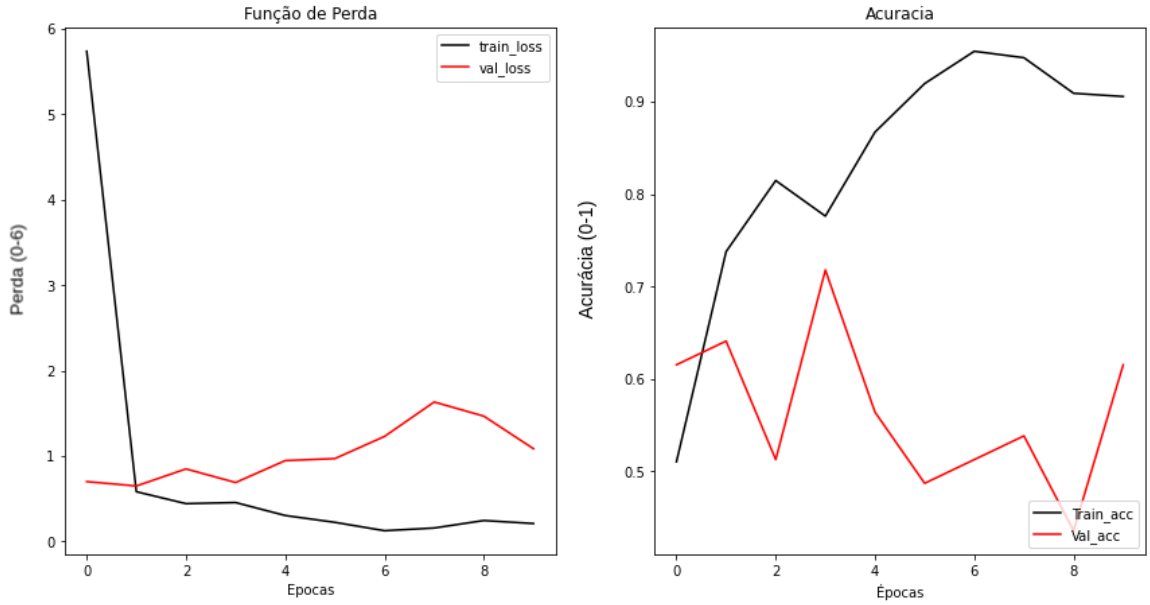
(b) CNN-3D

Figura 5 – Matriz de Confusão dos modelos. Fonte: elaborada pelo autor.

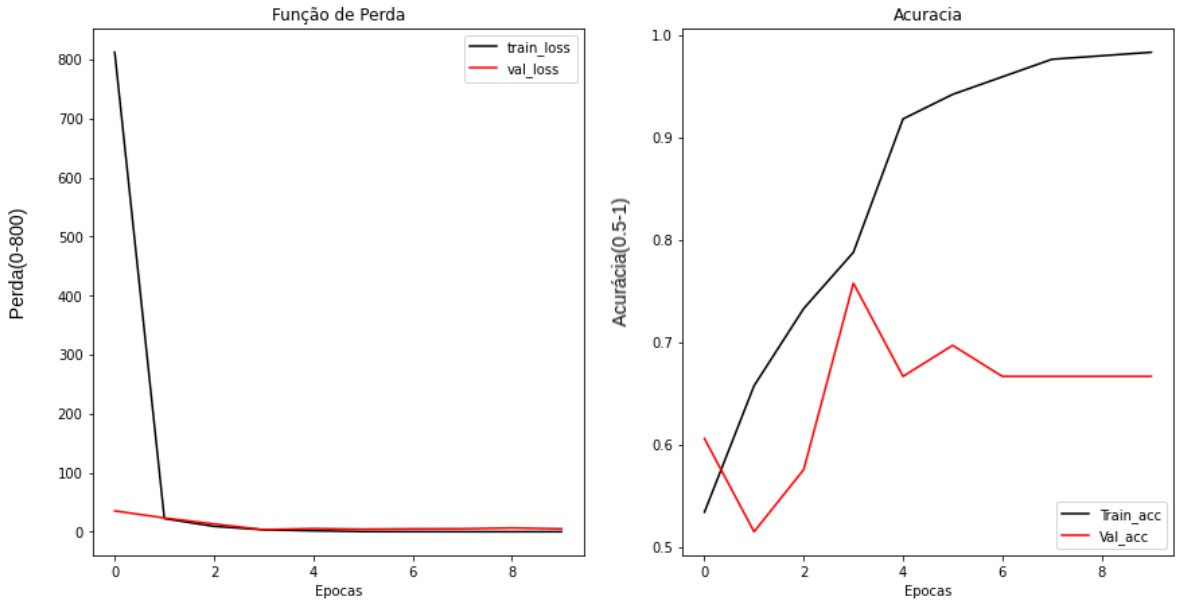
Durante a etapa de treinamento todos os modelos mostraram o comportamento de sobreajustes dos dados. Dessa forma o modelo perde a sua capacidade de generalizar para exemplos que ainda não foram vistos. A Figura 6 mostra como isso acontece em ambos os modelos. Nas imagens vemos que as funções de perda tanto da ConvLSTM e CNN-3D diminuem com o passar das épocas. Entretanto, com o passar das épocas vemos que a diferença entre a acurácia durante o treinamento e a de validação aumenta.

No caso da ConvLSTM, com o conjunto de treinamento obtemos uma acurácia 95%.

Já com o conjunto de validação, o melhor resultado foi de 72%. Para a CNN-3D, com o conjunto de treinamento obtemos uma acurácia de 98%. Com o conjunto de validação, o melhor resultado é de 75%.



(a) ConvLSTM



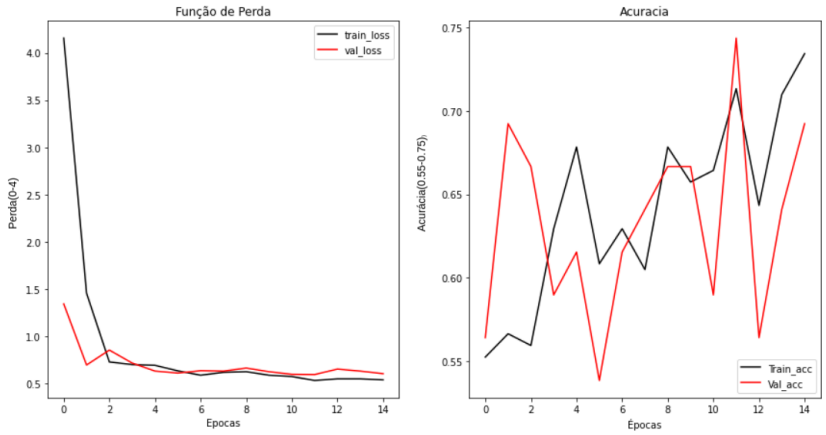
(b) CNN-3D

Figura 6 – Treinamento dos modelos. Fonte: elaborada pelo autor.

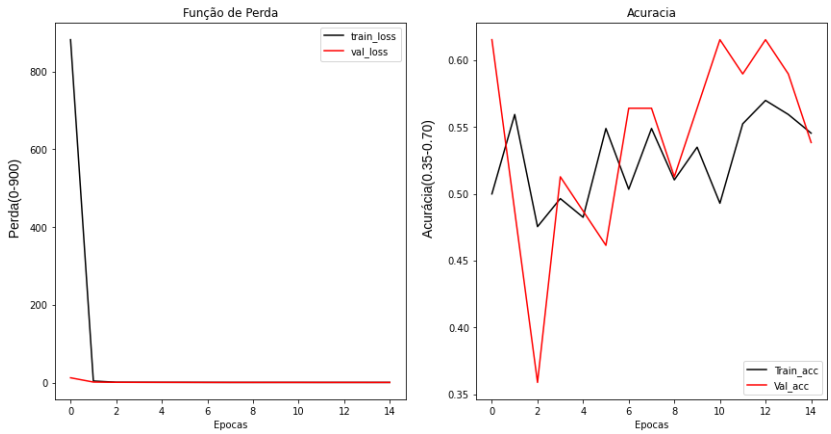
Uma maneira de evitar ou amenizar o sobreajuste do dados ao modelo é aplicar

alguma técnica de regularização. A mais comum é a de *dropout*. A Figura 7 traz o comportamento dos modelos quando submetidos à abordagem. No caso da ConvLSTM, para o conjunto de treinamento obtém uma acurácia de 73% e com o conjunto de validação temos 74%. Com a CNN-3D, o conjunto de treinamento alcançou 56% de acurácia e o com o conjunto de validação obteve 63%.

Como podemos ver na Figura 7, o comportamento dos modelos não muda muito em relação ao já apresentado na Figura 6 no que diz respeito à função de perda. Em relação à acurácia, vemos que o comportamento dos dados durante o treinamento é mais parecido com o que acontece na validação. Isso é diferente do que aconteceu com os modelos da Figura 6. Com isso, vemos que a aplicação consegue atenuar os problemas de sobreajuste ocasionados pelos modelos quando não aplicamos nenhum tipo de regularização.



(a) ConvLSTM



(b) CNN-3D

Figura 7 – Treinamento dos modelos. Fonte: elaborada pelo autor.

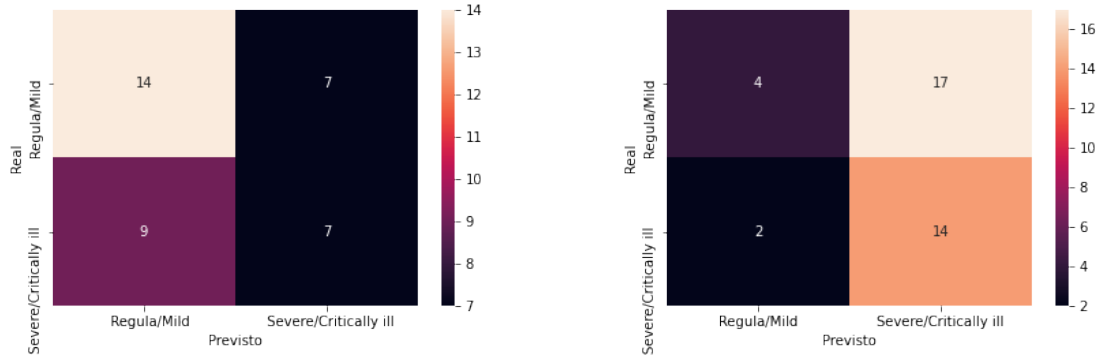
A correção do problema do reajuste acaba ocasionando uma queda no desempenho

acima dos padrões que seriam considerados satisfatórios tanta na ConvLSTM ,quanto na CNN-3D. Todas as métricas avaliadas são afetadas com aplicação de *dropout*. Isso fica mais visível na Tabela 4 que possui a comparação de desempenho entre os modelos.

Tabela 4 – Tabela de comparação dos resultados com o uso de dropout.

Modelo	Dropout	Acurácia	Precisão	Revocação	F1-score	AAC
ConvLSTM	sem	81%	89%	76%	82%	82%
CNN-3D	sem	73%	81%	65%	72%	74%
ConvLSTM	com	57%	50%	44%	47%	55%
CNN-3D	com	49%	45%	87.5%	59%	53%

Na Tabela 4 vemos que, além das quedas nas métricas, o *dropout* traz aos modelos um comportamento mais próximo aos de modelos aleatórios. Isso pode ser visto no resultado de AAC que ficam próximos a 50% que é o valor alcançado por modelos puramente arbitrário. Esse comportamento fica ainda mais visível quando vemos a Figura 8 que traz a matriz de confusão dos dois modelos.



(a) ConvLSTM (b) CNN-3D

Figura 8 – Matriz de Confusão dos modelos com dropout. Fonte: elaborada pelo autor.

Outro ponto que podemos notar na Figura 8 é que os modelos tem comportamentos diferentes. Enquanto que o ConvLSTM prevê mais amostras do tipo Regular/Leve, a CNN-3D classifica mais amostra do tipo Severo/Crítico.

Nos apêndices A, B, C e D temos os detalhes das arquiteturas e os códigos dos modelos de ConvLSTM e CNN-3D que alcançam os resultados destacados nesta seção.

5 CONCLUSÃO

Este trabalho busca explorar o uso de algoritmos de DL para fazer uma avaliação de gravidade de um paciente acometido de covid-19. Os dados utilizados seriam TC dos pacientes, que passariam por uma etapa de pré-processamento e depois passariam pelos algoritmos de classificação. Os algoritmos utilizados foram: ConvLSTM e CNN-3D. Conforme foi mostrado na seção 4, vimos quer o algoritmo de ConvLSTM foi o que obteve os melhores resultados.

Com base no que vimos na Seção 4, percebemos que é possível estabelecer uma avaliação de gravidade para casos de covid-19 com o uso de técnicas DL. Os algoritmos de ConvLSTM e CNN-3D obtiveram bons resultados, considerando que tinham uma arquitetura simples e a base de dados utilizadas ser pequena. Contudo, ainda estão longe dos resultados ideais, visto que as diferenças para os resultados de outro artigos foi maior do que a considerada satisfatória. Por isso, há necessidade de estudos mais aprofundados para chegar a resultados mais conclusivos sobre o potencial dos algoritmos.

Além disso, o problema de sobreajuste conseguiu ser atenuado, usando a técnica de dropout. Entretanto, acabou gerando uma queda muito grande no desempenho. Dessa forma, os objetivos de explorar os modelos e a avaliação das técnicas de regularização foram atingidos. Contudo, os resultados alcançados não foram satisfatórios.

Algumas dificuldades relacionadas a infraestrutura no ambiente de desenvolvimento em que foram realizadas os testes, não possibilitaram a exploração de técnicas mais complexas para resolver o problema. Para trabalhos seguintes, é necessário aumentar o conjunto de dados disponíveis seja de maneira artificial ou com outras bases. Também é importante que testes com modelos mais complexos possam ser feitos.

REFERÊNCIAS

ALAKWAA, W. **Lung Cancer Detection and Classification with 3D C**. [*S. l.*]: International Journal of Advanced Computer Science and Applications, 2018.

DABBURA, I. **Coding Neural Network — Dropout**. 2018. Disponível em: https://towardsdatascience.com/coding-neural-network-dropout-3095632d25ce.

DASTIDER FARHAN SADIK, S. A. F. A. G. **An integrated autoencoder-based hybrid CNN-LSTM model for COVID-19 severity prediction from lung ultrasound.** 2021.

FENG, Y.-Z.; LIU, S.; CHENG, Z.-Y.; QUIROZ, J. C.; REZAZADEGAN, D.; CHEN, P.-K.; LIN, Q.-T.; QIAN, L.; LIU, X.-F.; BERKOVSKY, S.; COIERA, E.; SONG, L.; QIU, X.-M.; CAI, X.-R. **Severity Assessment and Progression Prediction of COVID-19 Patients Based on the LesionEncoder Framework and Chest CT**. 2021.

GERON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow**. 1th. ed. [*S. l.*]: Editora Alta Books, 2019.

KHOZEIMEH DANIAL SHARIFRAZI, N. H. I. J. H. J.-A. S. R. A. J. M. G. S. H. Z. A. S. H. M. A. K. S. N. . S. M. S. I. F. **Combining a convolutional neural network with autoencoders to predict the survival chance of COVID-19 patients.** 2021.

LEI, L. W. S. **iCTCF - CT images and clinical features for Covid-19**. 2020. Disponível em: http://ictcf.biocuckoo.cn/Contact.php.

LI ZHENG ZHONG, Y. L. T. Z. L. G.-D. J. Y. S. X. Y. L. Y. Z. **From community-acquired pneumonia to COVID-19: a deep learning-based method for quantitative analysis of COVID-19 on thick-section CT scans**. 2020.

MOHAMMADI YINGXU WANG, N. E. P. A. F. N.-A. M. J. R. H. C. O. S. Y. K. N. P. A. **Diagnosis/Prognosis of COVID-19 Chest Images via Machine Learning and Hypersignal Processing**. 2021.

SAK ANDREW SENIOR, F. B. H. **Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling**. 2014.

SHI ZHOURONG CHEN, H. W. D.-Y. Y. W.-k. W. W.-c. W. X. **Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting**. [*S. l.*]: Neural Information Processing Systems (NIPS), 2015.

TAN, W.; LIU, J. **A 3D CNN Network With BERT for Automatic COVID-19 Diagnosis From CT-Scan Images**. 2021. 439-445 p.

APÊNDICE A – CONVLSTM

O modelo de ConvLSTM que é demonstrado na figura 9. Ele possui duas camadas de ConvLSTM com 32 filtros na primeira e 16 na segunda. Também possui uma camada de *pooling* de máximo com uma janela de 3x3x3. No final tem duas camada densa com 50 e 25 neurônios.

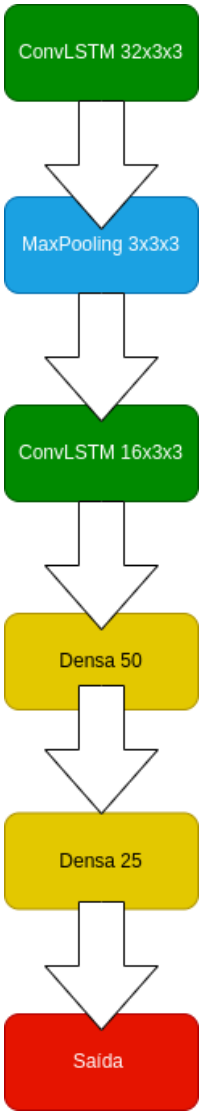


Figura 9 – Arquitetura da ConvLSTM. Fonte: elaborada pelo autor.

Para treinar foram utilizados a função custo de entropia cruzada e o otimizador Adam. Além disso, foram usado lote com um tamanho de 12 imagens e o treinamento foi feito em 10 épocas. A seguir temos o código:

```
import tensorflow as tf
```

```
batch_size = 12
epoch = 10

callback = tf.keras.callbacks.EarlyStopping(monitor = "val_accuracy",
                                            patience = epoch,mode="max",
                                            restore_best_weights = True)

model = tf.keras.Sequential()


model.add(tf.keras.layers.ConvLSTM2D(filters=32,
                                     kernel_size=[3,3],
                                     padding = 'same',
                                     data_format='channels_last',
                                     return_sequences = True,
                                     input_shape = (150,64,64,1)))

model.add(tf.keras.layers.MaxPooling3D(pool_size = (3,3,3),padding = 'valid',
                                       data_format = 'channels_last'))

model.add(tf.keras.layers.ConvLSTM2D(filters=16,
                                     kernel_size=[3,3],
                                     padding = 'same',
                                     data_format='channels_last',
                                     return_sequences = True))




model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(50,activation ='relu'))
model.add(tf.keras.layers.Dense(25,activation ='relu'))
```

```
model.add(tf.keras.layers.Dense(2,activation = 'softmax'))


model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer = tf.keras.optimizers.Adam(),
              metrics=['accuracy'])


history = model.fit(x=X,y=y,
                    validation_split=0.12,
                    steps_per_epoch=None,
                    shuffle=True,
                    epochs = epoch,
                    batch_size = batch_size,
                    callbacks = [callback]
                    )
```

APÊNDICE B – CNN-3D

O modelo de CNN-3D com melhor desempenho esta descrito na figura 10. Ele é composto por uma camada de CNN-3D e uma de *pooling* de máximo. No final, temos uma camada densa de 50 neurônios e outra de 25 neurônios.

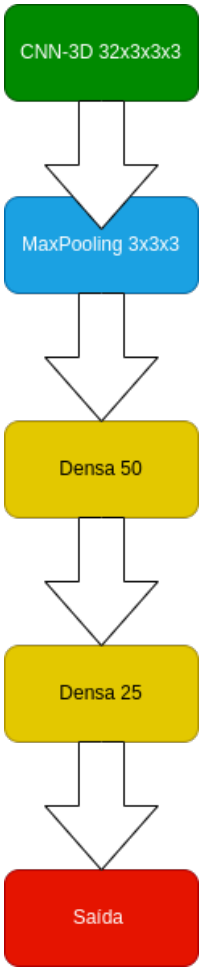


Figura 10 – Arquitetura da CNN-3D. Fonte: elaborada pelo autor.

Para o treinamento, a função de custo utilizada foi a de entropia cruzada e otimizador foi o Adam. O lote de tinha 12 imagens de tamanho e o treinamento teve 10 épocas. A seguir temos o código:

```
import tensorflow as tf

batch_size = 12
epoch = 10

callback = tf.keras.callbacks.EarlyStopping(monitor = "val_accuracy",
```

```
                                                patience = epoch,mode="max",
                                                restore_best_weights = True)

model_cnn = tf.keras.Sequential()
model_cnn.add(tf.keras.layers.Conv3D(filters = 32, kernel_size = [3,3,3],
                                     activation='relu',
                                     data_format = "channels_last",
                                     input_shape = (150,64,64,1)))

model_cnn.add(tf.keras.layers.MaxPooling3D(pool_size = (3,3,3),padding ='valid',
                                           data_format = 'channels_last'))


model_cnn.add(tf.keras.layers.Flatten())
model_cnn.add(tf.keras.layers.Dense(50,activation ='relu'))
model_cnn.add(tf.keras.layers.Dense(25,activation ='relu'))
model_cnn.add(tf.keras.layers.Dense(2,activation ='softmax'))

model_cnn.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer = tf.keras.optimizers.Adam(),
              metrics = ['accuracy'])

history_cnn = model_cnn.fit(x=X,y=y,
                  validation_split=0.12,
                  epochs = epoch,
                  batch_size = batch_size,
                  callbacks=[callback])
```

APÊNDICE C – CONVLSTM COM DROPOUT

O modelo de ConvLSTM segue o mesmo princípio do apresentado no apêndice A adicionando o *dropout* nas camadas *Dense*. A figura 11 mostra essa adição realizada e o modelo em seu estado final. Como vemos, as camadas de ConvLSTM e *Dense* possuem as mesmas características apresentadas no apêndice A.

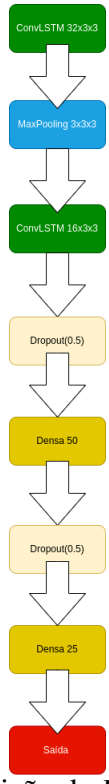


Figura 11 – modelo de ConvLSTM com adição de *Dropout*.Fonte: Elaborado pelo autor.

Para treinar foram utilizados a função custo de entropia cruzada e o otimizador Adam. Além disso, foram usado lote com um tamanho de 12 imagens e o treinamento foi feito em 10 épocas. A seguir temos o código:

```
import tensorflow as tf

batch_size = 12
epoch = 10

callback = tf.keras.callbacks.EarlyStopping(monitor = "val_accuracy",
           patience = epoch,mode="max",
           restore_best_weights = True)
```

```
model = tf.keras.Sequential()


model.add(tf.keras.layers.ConvLSTM2D(filters=32,
                                     kernel_size=[3,3],
                                     padding = 'same',
                                     data_format='channels_last',
                                     return_sequences = True,
                                     input_shape = (150,64,64,1)))

model.add(tf.keras.layers.MaxPooling3D(pool_size = (3,3,3),padding = 'valid',
                                       data_format = 'channels_last'))

model.add(tf.keras.layers.ConvLSTM2D(filters=16,
                                     kernel_size=[3,3],
                                     padding = 'same',
                                     data_format='channels_last',
                                     return_sequences = True))




model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(50,activation ='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(25,activation ='relu'))
model.add(tf.keras.layers.Dense(2,activation = 'softmax'))


model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
```

```
                optimizer = tf.keras.optimizers.Adam(),
                metrics=['accuracy'])


history = model.fit(x=X,y=y,
                    validation_split=0.12,
                    steps_per_epoch=None,
                    shuffle=True,
                    epochs = epoch,
                    batch_size = batch_size,
                    callbacks = [callback]
                    )
```

APÊNDICE D – CNN-3D COM DROPOUT

O modelo de CNN-3D ,apresentado na figura 12 segue o mesma ideia do apresentado no apêndice B. A camada de convolução e as camada *Dense* possuem as mesmas características do modelo mostrado B com a adição de *dropout* nas camadas *Dense*.

Figura 12 – CNN-3D com dropout. Fonte: elaborada pelo o autor

Para treinar foram utilizados a função custo de entropia cruzada e o otimizador Adam. Além disso, foram usado lote com um tamanho de 12 imagens e o treinamento foi feito em 10 épocas. A seguir temos o código:

```
import tensorflow as tf


batch_size = 12
epoch = 10

callback = tf.keras.callbacks.EarlyStopping(monitor = "val_accuracy",
                                            patience = epoch,mode="max",
                                            restore_best_weights = True)

model_cnn = tf.keras.Sequential()
```

```
model_cnn.add(tf.keras.layers.Conv3D(filters = 32, kernel_size = [3,3,3],
                                     activation='relu',
                                     data_format = "channels_last",
                                     input_shape = (150,64,64,1)))

model_cnn.add(tf.keras.layers.MaxPooling3D(pool_size = (3,3,3),padding ='valid',
                                           data_format = 'channels_last'))


model_cnn.add(tf.keras.layers.Flatten())

model_cnn.add(tf.keras.layers.Dropout(0.5))
model_cnn.add(tf.keras.layers.Dense(50,activation ='relu'))
model_cnn.add(tf.keras.layers.Dropout(0.5))
model_cnn.add(tf.keras.layers.Dense(25,activation ='relu'))
model_cnn.add(tf.keras.layers.Dense(2,activation ='softmax'))

model_cnn.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer = tf.keras.optimizers.Adam(),
              metrics = ['accuracy'])

history_cnn = model_cnn.fit(x=X,y=y,
                  validation_split=0.12,
                  epochs = epoch,
                  batch_size = batch_size,
                  callbacks=[callback])
```