Universidade Federal de Minas Gerais

Aplicações de Redes Neurais com Características de Enxame

Pesquisa Mista

Orientadora
Gisele Lobo Pappa

Luiz Paulo Santos Ribeiro

1. Introdução

Nos últimos anos, temos vivenciado um grande avanço nos algoritmos e ferramentas de AI. Seja no contexto de geração de imagens ou modelos de linguagem, redes neurais têm se mostrado uma tecnologia crucial para esses avanços. Entretanto, o uso extensivo de redes neurais profundas traz também seus desafios.

Esses modelos utilizam grande quantidade de parâmetros para alcançarem seus níveis de precisão, precisando também serem treinados numa quantidade extensa de dados. Esses fatores geram altos custos, tanto em termos computacionais,quanto em termos econômicos devido ao custo elevado para treinar esses modelos. Por referência, o modelo GPT-4 da OpenAI chega a quase 1.8 trilhões de parâmetros e mesmo modelos concorrentes que buscam ser mais leves ainda não escapam da ordem dos bilhões de parâmetros (Llama 3.2, modelo de código aberto da Meta recentemente chegou em modelos de 1 e 3 bilhões de parâmetros), enquanto o modelo Gemini Ultra da Google é estimado ter custado cerca de 200 milhões de dólares para treinar seus 500 bilhões de parâmetros.

Formas de miniaturizar esses modelos são muito cobiçadas, por criarem a possibilidade de novos usos locais para redes neurais, que no momento dependem largamente de servidores com hardware industrial para processamento e armazenagem massiva de dados. Modelos menores abrem a possibilidade para que usuários possam treinar suas instâncias locais ao seus problemas específicos, sem as dificuldades e riscos associadas à dependência de uma solução em nuvem.

Uma possibilidade para reduzir esses modelos é utilizando características de um enxame para redes neurais, como proposto por Nguyen Ha Than e Nguyen Le Minh, utilizando um "filtro" para extrair características de um enxame para a rede neural.

Os filtros de enxame são descritos como vetores $S = \{s_0, s_1, s_2, ..., s_m\}$ que opera sobre o vetor de entrada $X = \{x_0, x_1, x_2, ..., x_n\}$, utilizando a média dos n produtos externos referentes a cada instância de $s_j \in S$ para formar um vetor de m dimensões Y:

$$Yj = 1/n * \sum_{i}^{n} (X \bullet S)ji$$

No seu artigo, eles foram capazes de criar um modelo SCNN (Swarm Characteristic Neural Network) que foi capaz de performar melhor que os modelos MLP (Multi Layer Perceptron), CNN (Convolutional Neural Network) e LSTM (Long Short-Term Memory) usados na comparação, mesmo tendo apenas uma fração dos seus parâmetros.

Outros comportamentos desejados foram também identificados. O modelo SCNN apresentou convergência mais rápida que os demais modelos. Ademais, a extração de características de enxame, que pode ocorrer em uma parte arbitrária da rede, é uma janela para interpretar os resultados e processos internos de uma rede neural, muitas vezes criticadas por não possuir formas convencionais de visualizar o seu processo de decisão para os dados analisados.

O artigo, porém, se mostrou limitado em sua exploração, testando o conceito apenas contra um único caso de uso. Esse projeto tem como objetivo expandir nessa abordagem, inicialmente replicando o modelo do artigo e em seguida explorando outras aplicações de redes neurais para comparar a performance do modelo SCNN. Adicionalmente, iremos trazer novos parâmetros para essa

comparação que não foram utilizados anteriormente, como tempo de execução, já que é questionável se o peso de resolver os filtros de enxame desfaz significativamente os ganhos atrelados a redução do número de parâmetros.

Durante a primeira etapa de nosso projeto orientado, foi realizada a reprodução do experimento de Nguyen com o objetivo de verificar os achados e replicar a tecnologia das redes de enxame. Apesar da falta de especificidade das condições de teste, fomos capazes de chegar a resultados similares, com uma diferença condizente com uma parametrização distinta. Aqui, porém, a SCNN se mostrou ligeiramente menos precisa que a rede LSTM, com uma acurácia no set de validação de apenas 80% (contra 82% na LSTM, 79% para a CNN e 76% para a MLP).

Fomos ainda capazes de medir outros aspectos nos quais as redes de enxame se sobressaíram em relação às demais. O tempo de execução e treinamento da rede SCNN se mostrou apenas ligeiramente maior que a rede mais simples dos testes (MLP), com cerca de 240 batches por segundo nas configurações de teste (contra 255 na MLP, 70 na CNN e apenas 30 na LSTM). Notou-se também uma qualidade inesperada: analisando a acurácia no set de treino e de teste, observou-se que a rede SCNN se mostrou a mais resiliente a overfitting, com o distanciamento entre os sets crescendo de forma mais lenta que os demais modelos testados.

Dado esse sucesso na etapa anterior, prosseguiremos em expandir os cenários de testes usados para avaliar as SCNN, já que até o momento presente, ela apenas foi utilizada para resolver o problema de análise de sentimento de mensagens textuais, que se caracteriza como um problema de classificação binário e balanceado. Essa limitação cria questionamentos a respeito da capacidade dos filtros de enxame possuírem a flexibilidade e robustez necessária para abordar outros problemas.

2. Metodologia

Foram selecionados três problemas para que possamos comparar a performance das redes SCNN com três outras arquiteturas comuns de redes neurais, as redes de múltiplas camadas de perceptrons (Multi-Layer Perceptron ou MLP), as redes convolucionais (CNN) e as redes de memória de curto e longo prazo (Long Short Term Memory ou LSTM).

Os três problemas selecionados buscam se diferenciar em relação àquele abordado no artigo original e nossa reprodução da tecnologia, que testou o problema de classificação sentimental de texto e pode ser generalizado como um problema de classificação binária e balanceada. Assim foram selecionados um problema de classificação desbalanceada, um outro de classificação multi-classe, e por fim um problema de regressão numérica. Desse modo, problemas específicos selecionados foram então detecção de fraude em cartão de crédito, classificação de flores por imagem, e previsão de preço de ações no mercado financeiro.

Algumas características de nossa implementação são mantidas em todos os modelos e problemas para evitar discrepâncias em nossos resultados, entre elas mantivemos uma taxa de aprendizado fixa de 0.001, o uso do otimizador adams para o aprendizado, e a implementação em pytorch dando preferência a implementações internas para uma maior consistência. A função de perda usada é a perda entrópica para as classificações e erro quadrático médio para nossa regressão. Todos os modelos foram treinados em CPU (Ryzen 9 7950x, 16-cores) com 32 GB de memória RAM disponível.

2.1 Classificação Desbalanceada

Aqui utilizamos um banco de dados que nos informa 285 mil transações, das quais apenas cerca de 500 estão catalogadas como fraudulentas e as demais como genuínas. Os atributos desse banco de dados largamente sofreram transformações para proteger a privacidade dos usuários, com apenas o momento da transação e o seu valor permanecendo inalterado, mas outros 28 atributos também são fornecidos. O baixo número de positivos na nossa amostra nos levou a optar por oversampling as fraudes no grupo de treino.

Todos os modelos são compostos de uma camada intermediária simples conectada aos 30 atributos de entrada para abstração, e terminam com uma conexão simples a camada de saída com dois atributos. Essa modelagem básica é a inteirada da rede MLP. Já as demais são construídas pela introdução de suas respectivas arquiteturas entre a camada intermediária e a camada de saída.

2.2 Classificação Multiclasse

Aqui foi utilizado um banco de dados de cerca de 4.300 imagens de flores divididas cinco tipos de flores, cada qual representa cerca de 20% amostras: margaridas (18%), dentes-de-leão (24%), rosas (18%), girassóis (17%), tulipas (23%). Se tratando de imagens, realizamos uma normalização, resultando em imagens compostas de 3 canais de cor de 256x256 pixels, sendo o valor do pixel também normalizado.

O modelo MLP é construído de maneira simples, conectando as entradas a uma camada intermediária e então a saída. A rede CNN foi feita com uma camada convolucional, seguida de maxpooling, outra convolucional que se conecta diretamente com a saída. A LSTM trata a imagem como uma sequência de colunas a serem analisadas por um módulo LSTM que então se conecta à saída. Por fim, a rede SCNN usa uma camada intermediária seguida de dois filtros de enxame consecutivos que então conectam a saída.

2.3 Regressão Numérica

Restringimos o escopo de nossa previsão ao próximo dia de exchange da ação, baseado nos 31 dias úteis anteriores, utilizando os mesmos atributos que estamos tentando prever: o preço de abertura, fechamento, máxima, mínima e o volume total de transações. Os dados para essa previsão são extraídos em tempo real pela ferramenta de finanças do Yahoo!, especificamente usando o valor da empresa Google ao longo dos últimos 3 anos para prever os últimos 30 dias de exchange registrados. Os atributos não mencionados são descartados e os restantes são normalizados para a análise estatística e treino, porém retornados para sua escala original para a visualização do resultado.

A entrada de todas as redes é composta de 5 atributos distintos através dos 31 dias de dados. A MLP novamente usa uma camada intermediária conectada diretamente à saída. A CNN usa uma camada convolucional unidimensional, que é aplicada apenas a um mesmo atributo sob os dias, seguido então de maxpooling e outra convolucional antes da saída. A LSTM usa um módulo conectado aos 5 atributos para processar a sequência de dias. Por fim, a SCNN usa uma camada intermediária seguida de 2 filtros de enxame antes da saída.

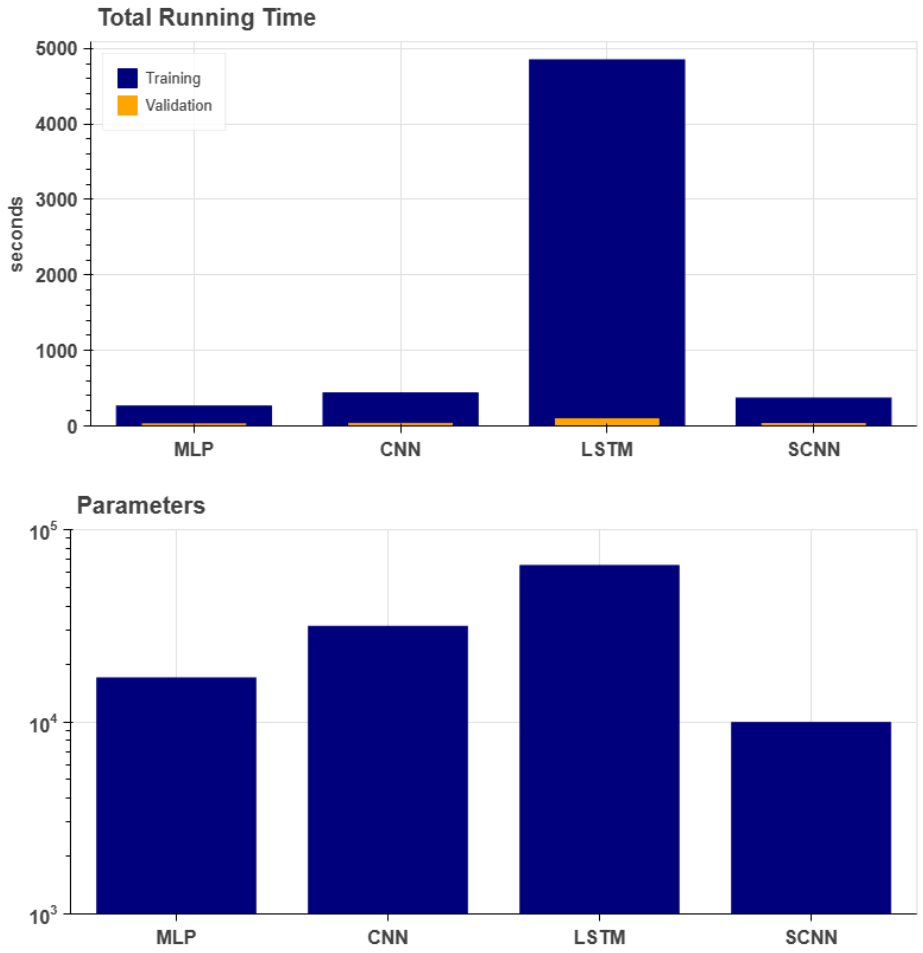
3. Resultados Experimentais

Iremos analisar a performance das redes SCNN, em relação aos demais modelos, em cada um dos problemas abordados com base em alguns critérios gerais. Primeiramente, estamos preocupados com o uso de recursos computacionais pelas redes, e para isso temos como parâmetros principais o tempo de treino e o número de parâmetros utilizados pelas SCNN em relação aos demais modelos. Neste quesito teremos uma análise holística dos modelos, enquanto o artigo original ignora um grande bottleneck comum a todos os modelos que foi a camada de embedding que dominava o número total de parâmetros por mais de uma ordem de magnitude.

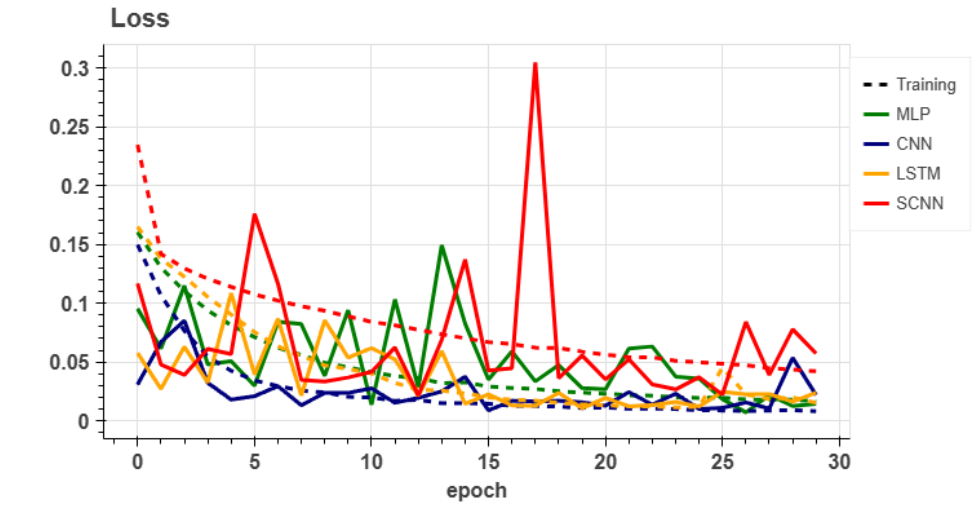
Em segundo lugar, nos preocupamos em preservar a acurácia das nossas previsões. Porém como cada um dos problemas abordados é consideravelmente distinto entre si, focamos-nos na função de perda e na tendência ao overfitting, aqui considerada como a tendência entre a perda de treino se afastar daquela encontrada durante a validação. Outros parâmetros serão discutidos baseados na especificidade do problema.

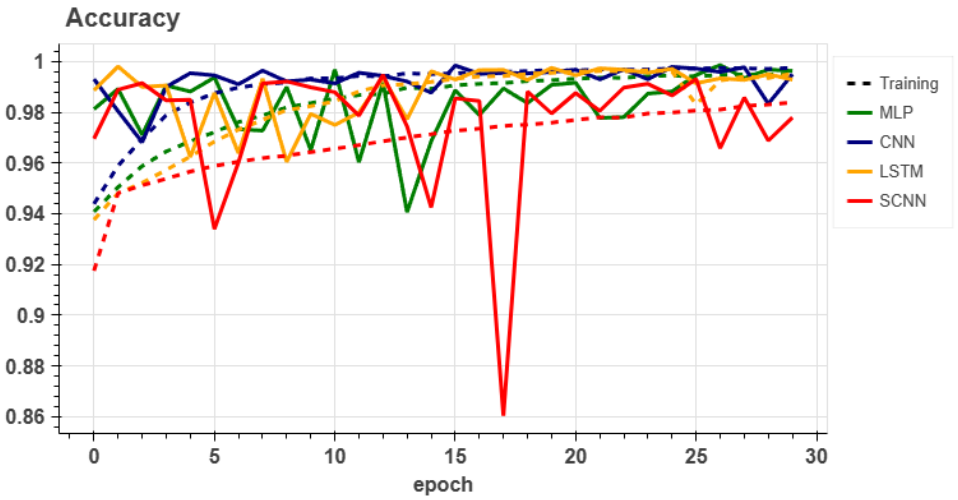
3.1 Classificação Desbalanceada

Obtivemos uma redução espacial significativa, o modelo SCNN utilizou apenas cerca de 10 mil parâmetros, contra 17 mil na rede MLP, 32 mil para a CNN e 64 mil para a rede LSTM. Já temporalmente vemos um resultado intermediário, cerca de 6 minutos para a rede SCNN, contra 4 na MLP, 7 na CNN e 81 minutos para a LSTM.

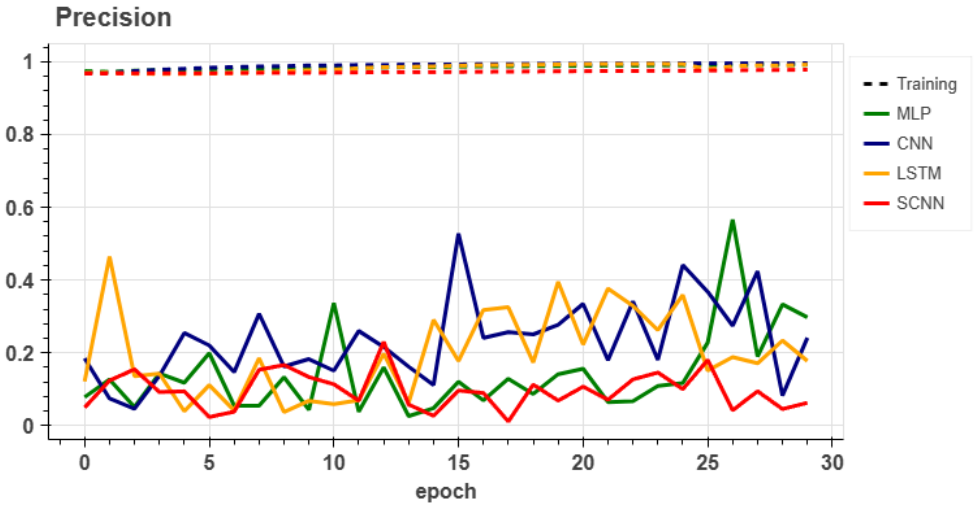


Mesmo com oversampling no treino, houve pouca tendência de overfitting detectada pela diferença na função de perda, porém isso é um se atrela a natureza do problema, havendo muito poucos positivos na validação para que esses impactem significativamente na perda e acurácia do modelo.

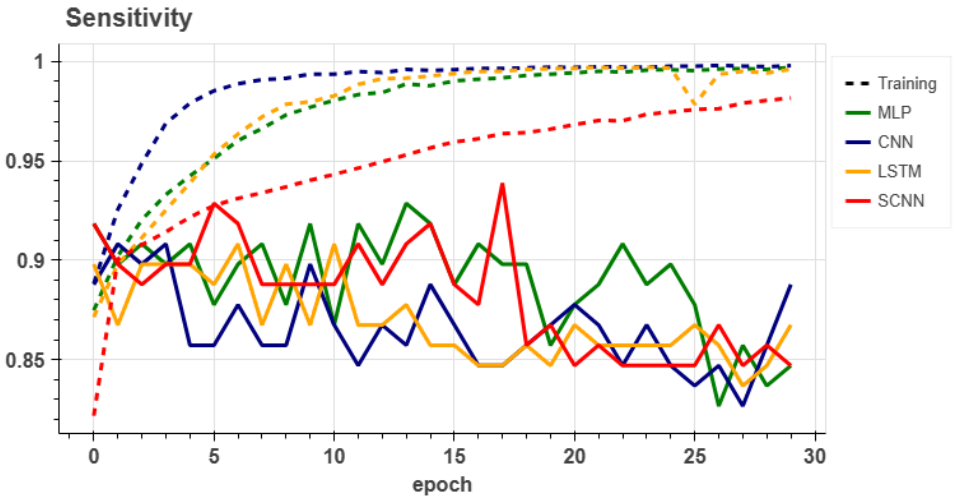




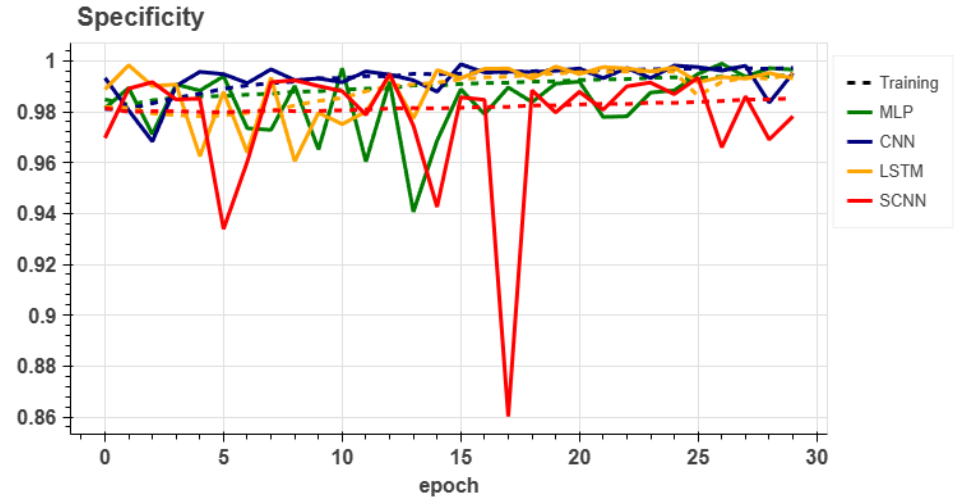
Nos movendo para métricas mais relevantes, vemos que a precisão dos modelos permanece alta durante o treino (cerca de 97% já na primeira época e com clara tendência de crescimento), porém muito mais baixa e errática durante a validação. A melhor precisão obtida ao longo das épocas foi de apenas 24% na SCNN, comparado com a MLP de 56%, 53% para a CNN e 47% na rede LSTM.



Em relação a sensitividade dos modelos, vemos alguma resistência ao overfitting por parte da SCNN, sendo essa também a única métrica que conseguiu performar acima dos demais modelos, com 94%, contra 93% na MLP e 91% nos demais modelos durante a validação.

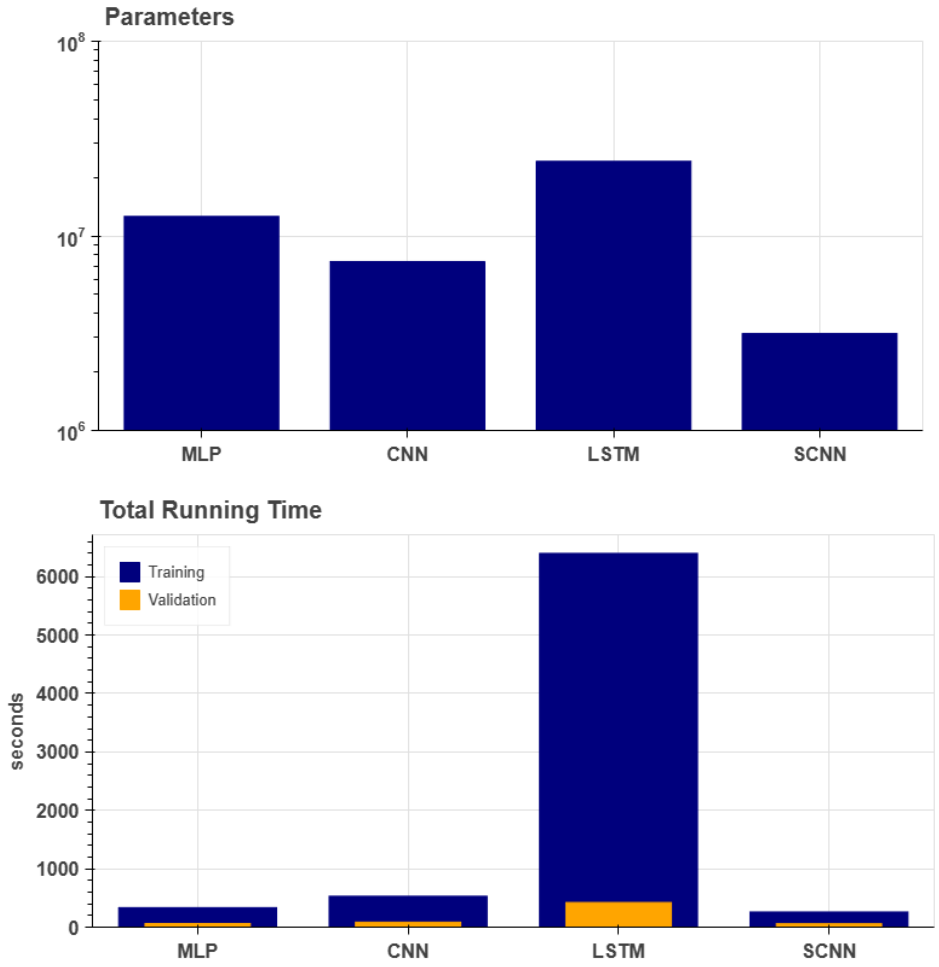


Para a especificidade vemos novamente um crescimento durante o treino e um comportamento erratico para a validação dos modelos, com todos os modelos obtendo valores altos. Nota-se porém que a rede SCNN é significativamente mais errática que as demais.

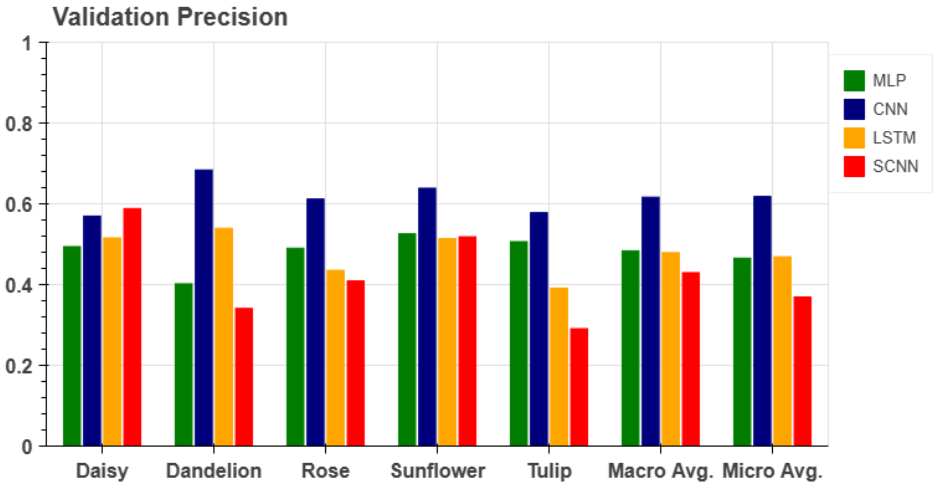


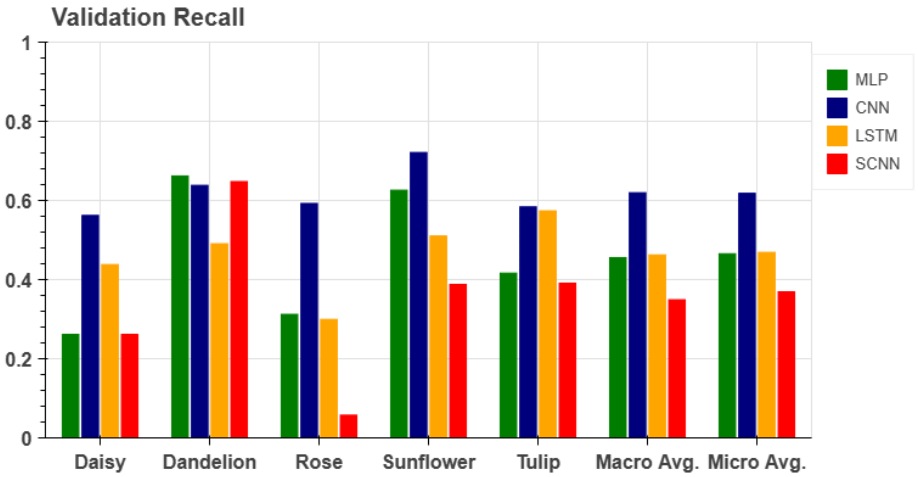
3.2 Classificação Multiclasse

Entre os modelos de classificação de flores, obtivemos resultados mais fortes em relação aos recursos computacionais. A SCNN foi construida com apenas 3 milhões de parametros, contra 7 na rede CNN, 12 na MLP e 24 na LSTM, e ordem similar se manteve temporalmente, com a SCNN sendo treinada em cerca de 4 minutos, 5 para a MLP, 9 para a CNN e 160 para a LSTM.



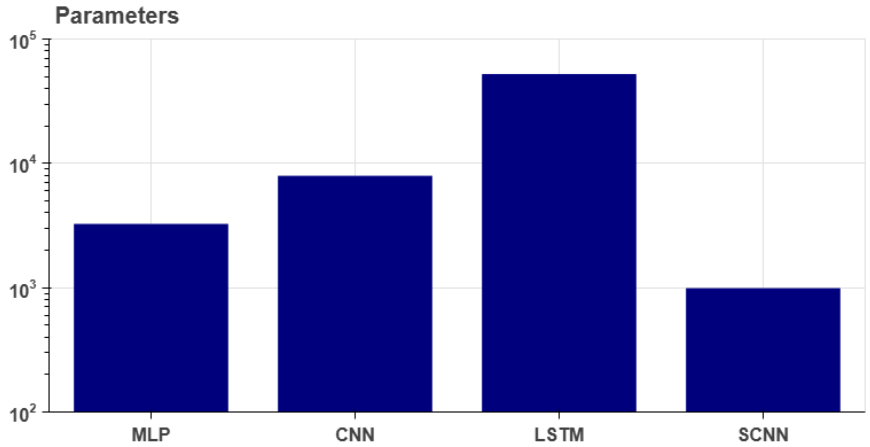
Vendo a evolução da função de perda, vemos que a SCNN mostrou resistência ao overfitting, similar a rede LSTM. Porém em relação às métricas, vemos resultados subpar, com uma performance geral significativamente abaixo das demais tanto em precisão quanto em recall durante nossa validação ao fim do treino. Mesmo com o overfitting a rede CNN atingiu uma acurácia de 62% (o que não é surpreendente, a literatura indica seu uso para problemas envolvendo análise de imagens), enquanto a rede SCNN obteve apenas 37%, e ambas MLP e LSTM, 47%. Observamos também que dentre os modelos, a SCNN mostrou o resultado menos consistente entre diferentes classes. Abaixo podemos ver melhor a precisão e recall de cada modelo com cada classe.



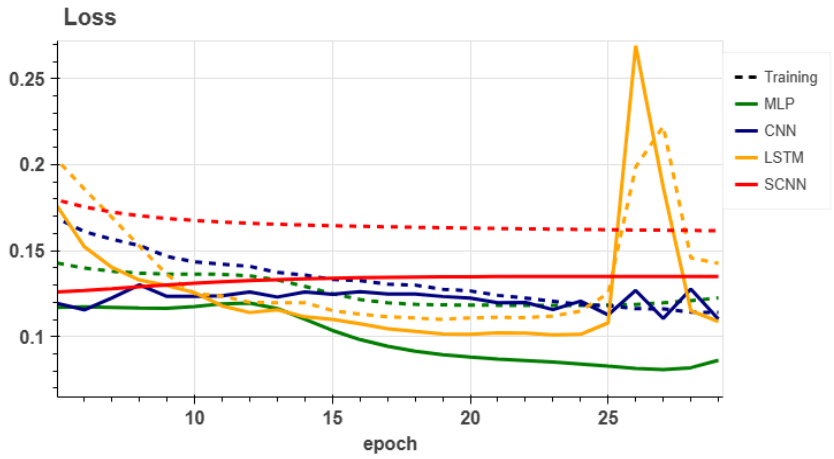


3.3 Regressão Numérica

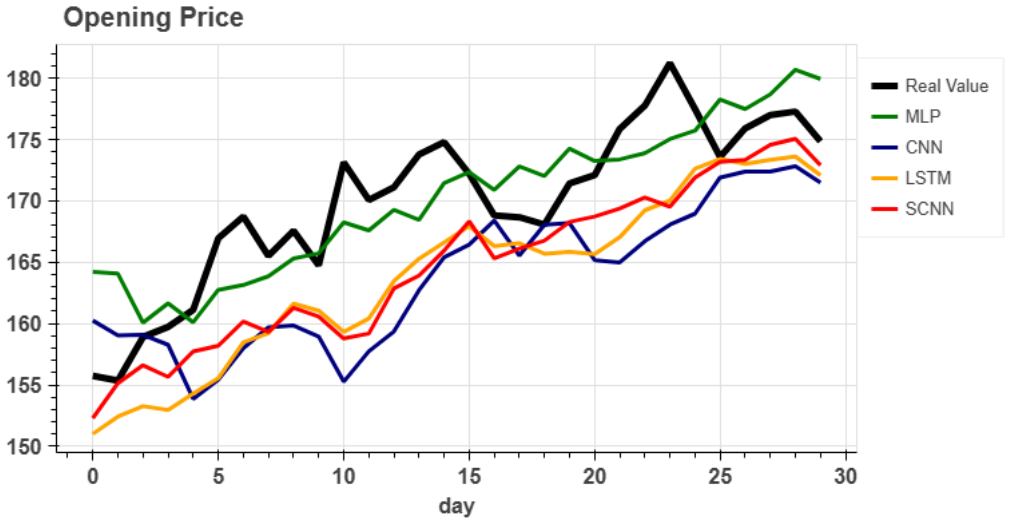
Mais uma vez obtivemos uma redução espacial significativa, a SCNN pode ser construída com apenas mil parâmetros, enquanto as redes MLP, CNN e LSTM utilizaram 3, 8 e 52 mil respectivamente. Temporalmente, este problema possui um tamanho negligenciável (menos de 800 amostras com 5 atributos cada), tomando meros 2 segundos para treinar as redes SCNN e CNN, 1 para a MLP e 19 com a rede LSTM.

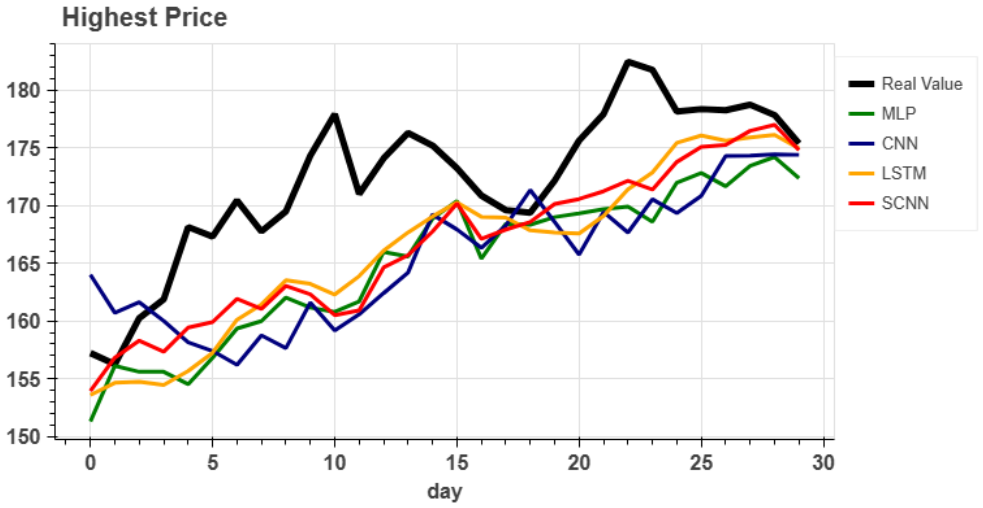


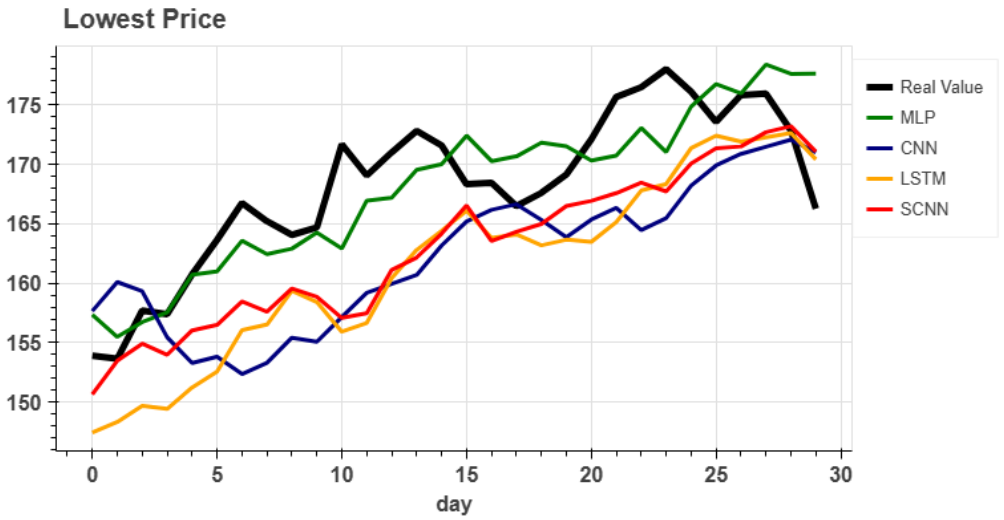
Com a função de perda vemos uma convergência de todos os modelos tanto durante o treino quanto durante a validação dos dados (últimos 30 dias de exchanges).

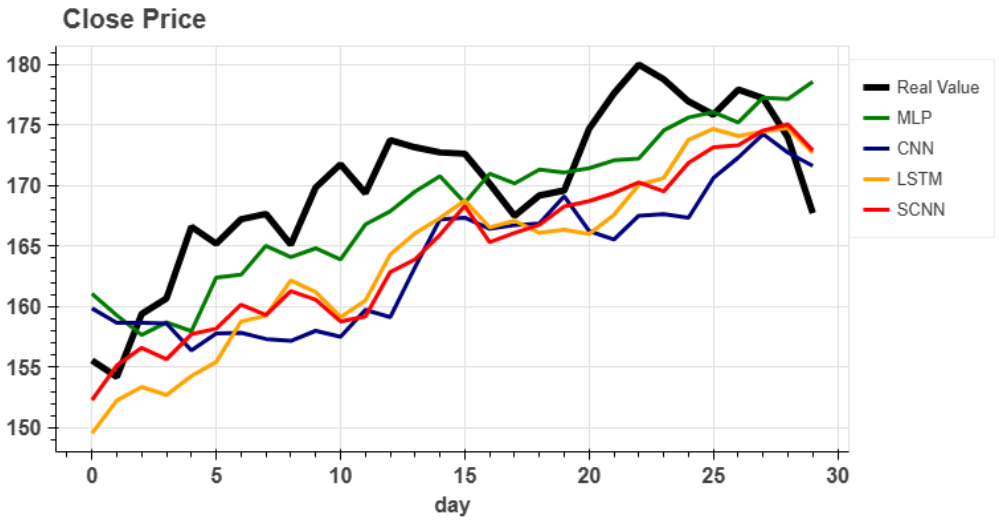


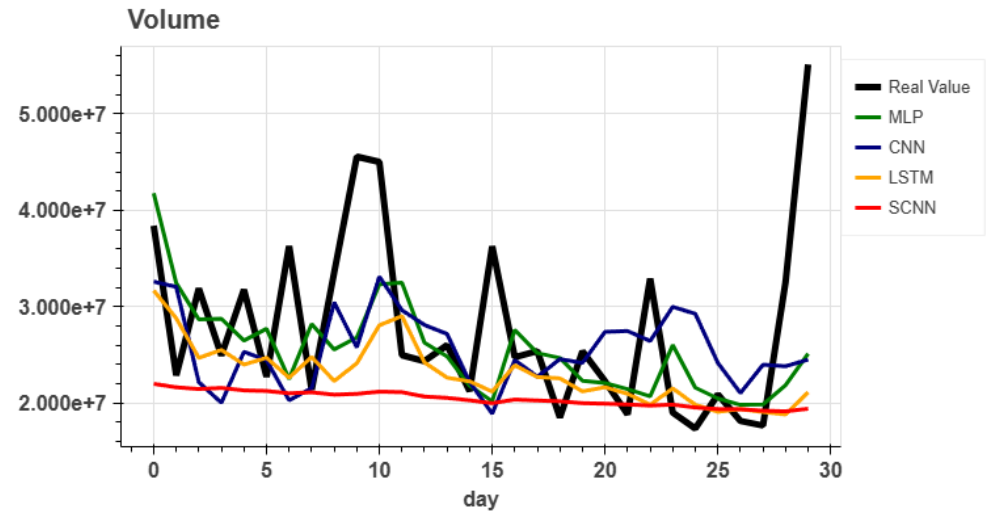
Apesar de que a perda indica um resultado abaixo dos demais modelos, analisando as previsões feitas vemos que a SCNN fez previsões competitivas com demais modelos, a exceção do volume de transações na qual performou abaixo das demais. Outra característica notável é que suas previsões são muito consistentes entre os 4 valores de preço a serem previstos.











4. Conclusões

De forma geral, vimos uma performance subpar das redes SCNN tanto em nosso problema de classificação desbalanceado, quanto na multiclasse. Uma forma de interpretar essa discrepância, considerando a alta performance da arquitetura na análise sentimental é olhando para as diferentes entradas de cada um desses.

Enquanto todos os problemas abordados aqui tem como entradas atributos contínuos, o processo de normalização que entradas textuais passam envolve etapas de tokenização, indexação e vetorização, onde cada token é convertido num vetor de atributos próprios e independentes entre si, os quais podemos interpretar como a posição da partícula no espaço n-dimensional. Já nas entradas contínuas, não há como criar tal vetorização independente de cada atributo, e ao invés disso criamos uma camada de abstração utilizando os atributos e pesos, resultando numa camada cujo valor médio é em si a média do produto entre os atributos e a média pesos associados àquele atributo.

Apesar da arquitetura ter sucedido em reduzir expressivamente o número de parâmetros em todos os problemas, assim como manter o custo temporal consistentemente abaixo da rede CNN e LSTM, vemos que a perda de informação resultante da compressão prejudicou fortemente a capacidade da rede em solucionar os problemas abordados, apenas se equiparando na regressão numérica.

Acreditamos que o próximo passo para está tecnologia esteja em retornar a problemas conceitualmente mais próximos do original, como classificação de tópico de artigos, ou detecção de spam em emails, dois cenários de entrada textual (multiclasse e desbalanceado, respectivamente) e que portanto compartilham da estrutura possivelmente favorável de tokenização e vetorização. Outra direção possível está na exploração de redes híbridas, que usam da arquitetura SCNN em paralelo com modelos mais consagradas na literatura e no mercado, de modo a buscar alternativas que gerem uma melhora das soluções que não impliquem tão somente no uso de redes maiores e mais complexas.

Referências Bibliográficas

Artigo referenciados

Nguyen Ha Thanh, Nguyen Le Minh. SCNN: Swarm Characteristic Neural Network. Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan 2021.

Nguyen Ha Thanh, Nguyen Le Minh. "Swarm Filter - A Simple Deep Learning Component Inspired by Swarm Concept". *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, Portland, OR, USA, 2019, pp. 1541-1545.

Banco de dados utilizados

Credit Card Fraud
Flower Dataset
Yahoo Finances

Repositório do com os problemas implementados e a reprodução do paper original

https://github.com/LuizPSR/POC

Dados a respeito do problema comentado

https://codingscape.com/blog/most-powerful-llms-large-language-models
https://medium.com/@georgeanil/visualizing-size-of-large-language-models-ec576caa5557
https://www.visualcapitalist.com/training-costs-of-ai-models-over-time/
https://huggingface.co/meta-llama/Llama-3.2-3B