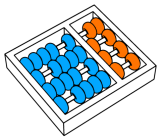

Modelo convolucional baseado em redes *Kolmogorov-Arnold* para classificação de imagens

F. G. Brabes *E. L. Colombini*

Relatório Técnico - IC-PFG-25-32
Projeto Final de Graduação
2025 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Modelo convolucional baseado em redes *Kolmogorov-Arnold* para classificação de imagens

Felipe Gabriel Brabes da Silva* Esther Luna Colombini*

Resumo

O projeto apresenta uma nova abordagem para a implementação de um modelo não-linear de convolução, inspirado nas redes Kolmogorov-Arnold. Neste estudo, também é feita a discussão de técnicas de regularização para essa camada de convolução e a análise das alterações que isso gera em seu desempenho para tarefas de classificação utilizando o dataset CIFAR10. A abordagem discutida, além de acelerar o treinamento e inferência desses modelos, destaca sua capacidade de aprendizagem local no espectro. De forma a analisar essa característica, foram feitos testes com diferentes níveis de distorção no espectro das imagens de treino e teste. Os resultados inclinam-se a afirmar a existência dessas capacidades, assim como a destacar a melhoria em desempenho proveniente das técnicas de regularização introduzidas.

1 Introdução

A classificação de imagens é um problema computacional que representa um obstáculo central em inúmeras aplicações que impactam diretamente a sociedade: de sistemas de diagnóstico médico assistido a ferramentas de monitoramento ambiental, essa tarefa é de grande importância e têm diversas aplicações. Melhorar o desempenho dos modelos responsáveis por essa tarefa significa oferecer decisões mais precisas e rápidas, ampliando a confiabilidade de tecnologias utilizadas por profissionais e usuários finais.

Problemas desafiadores, como a identificação precoce de doenças em exames de imagem, o reconhecimento de padrões em imagens de satélite e a detecção automática de anomalias em linhas de produção industrial, exemplificam a complexidade e relevância desse domínio. Os algoritmos criados não eram suficientemente robustos para esse problema, no qual os modelos de aprendizado profundo (*deep learning*) apresentaram uma quebra de paradigma, demonstrando bom desempenho em aplicações para diversos domínios de imagens.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

No entanto, apesar dos inúmeros avanços nas últimas décadas, modelos de classificação ainda apresentam dificuldades significativas ao lidar com variações extremas de luminosidade, especialmente em condições de baixa ou alta iluminação, o que limita sua adaptabilidade e robustez. Nesse contexto, este trabalho investiga um novo modelo de convoluções projetado para lidar melhor com tais cenários, visando ampliar a eficiência e a generalização em sistemas de classificação de imagens.

2 Trabalhos Relacionados

Antes do aprendizado de máquina, os algoritmos para realizar a tarefa de reconhecimento de imagens eram dominados por técnicas de extração e agregação de *features*, que utilizavam heurísticas determinadas por humanos para classificar as imagens. Embora os algoritmos que utilizassem essas heurísticas fossem os mais comuns, eles apresentam severas limitações nessa tarefa, e seu desempenho não era satisfatório para aplicações reais.

O campo de reconhecimento de imagens foi onde os algoritmos de aprendizado profundo se destacaram: O treinamento de modelos para a realização dessas tarefas com o uso de grandes coleções de dados possibilitava a extração das *features* e heurísticas automaticamente, adaptando-se especificamente ao domínio do problema e obtendo resultados muito melhores do que com as técnicas anteriores.

2.1 Redes neurais convolucionais (CNNs)

Durante a história das aplicações de *Deep Learning* para essa finalidade, se destaca o artigo [5] por introduzir as redes convolucionais neurais (CNNs) que passaram a dominar o campo na década de 90 e são referência até hoje. Essa arquitetura introduz *kernels* "aprendíveis"com o uso de *backpropagation*, possibilitando a modelos explorarem a extração de características das imagens com base na interação de pixeis localmente próximos, aproveitando-se da informação espacial inerente ao domínio.

Desde então, diferentes arquiteturas estado-da-arte surgiram para alavancar o uso das camadas convolucionais para a resolução desse problema, consolidando sua efetividade em gerar representações de mais alto nível das imagens. Atualmente, destaca-se a arquitetura das Vision Transformers (ViTs) [4] no estado-da-arte para classificação de imagens, uma arquitetura que faz uso de uma camada convolucional para extração de informações da imagem em uma representação que abastece as camadas de atenção da *transformer*.

2.2 Redes Kolmogorov-Arnold (KANs)

As Redes Kolmogorov-Arnold (KANs), introduzidas em [6], são uma arquitetura recente inspirada diretamente no teorema de superposição de Kolmogorov-Arnold. Esse teorema afirma que qualquer função contínua multivariada pode ser representada como uma soma finita de funções contínuas de uma única variável, compostas e somadas em uma estrutura predeterminada.

Diferentemente das redes MLP tradicionais, onde há combinações lineares seguidas de ativações fixas nos neurônios, as KANs deslocam a aprendizagem para as funções de ativação nas arestas (conexões): cada "peso"em uma KAN não é apenas um escalar fixo, mas sim uma função univariada aprendível, originalmente parametrizada por splines.

Além da capacidade de composição de funções não-lineares aprendíveis, otimizando o ajuste dos modelos de aprendizado de máquina para funções de ativação específicas da necessidade do domínio, se destaca outra capacidade dessa modelagem: o aprendizado local. Como essas funções são modeladas por splines, em determinada passagem pela rede (*feedforward*), apenas alguns pontos das splines, que são os parâmetros, serão ativados. Dessa maneira, durante a descida do gradiente (*gradient descent*) apenas uma parte da função é otimizada, mantendo seu comportamento inalterado para uma região do domínio da função.

Essa característica das splines na modelagem não foi explicitamente explorada pelo artigo original, embora tenha sido mencionada pelos autores.

2.3 Redes Convolucionais Kolmogorov-Arnold (KANCs)

Os princípios por trás das redes Kolmogorov-Arnold também foram aplicados à convolução por [2]. No artigo, os autores descrevem convoluções utilizando funções não-lineares no lugar de pesos de kernel fixos, como nas CNNs originais. A modelagem de cada elemento no kernel é descrita pela equação:

$$\phi = w_1 \cdot \text{bspline}(x) + w_2 \cdot \text{silu}(x) \tag{1}$$

Essa modelagem traz algumas desvantagens, dentre elas vamos destacar o tempo de treinamento: Segundo os autores, a implementação das KANs por B-Splines no kernel inviabiliza a paralelização com GPUs do treinamento, limitando as aplicações desse modelo em problemas do mundo real. Essa modelagem considera as splines de base (B-Splines) como necessárias para garantir que a função modelada será suave por construção, assegurando a otimização do algoritmo.

A soma com a função $\text{silu}(x)$ tem o objetivo de mitigar a falta de controle da função

quando o elemento do domínio está fora dos pontos compreendidos pelo grid que as B-Splines definem. Esse acontecimento apresenta uma grande dificuldade para o treinamento dos modelos KANs, mas assumir a frequência desse caso e delegar à silu para lidar com esses casos limita muito a capacidade das KANCs de modelar adaptativamente a não-linearidade.

2.4 Suavização de splines na modelagem estatística

Contudo, embora muito promissoras, as pesquisas sobre as redes KANs estão em estágio inicial, e por isso essas redes não possuem muitas das ferramentas consolidadas de aprendizado de máquina que aparecem em outros estilos de modelagem mais tradicionais, como regularizações. O paralelo dessas redes com a modelagem estatística por splines é evidente, em [3], Craven, et al. apontam como esses modelos apresentam, sem o uso de regularizações, uma tendência ao overfitting.

A função resultante apresenta um grau de ondulação muito grande, o que dificulta a generalização desse tipo de abordagem. Para contornar essa barreira apresentada pelas então chamadas *wiggly functions*, buscando uma modelagem mais robusta, os autores propoem um fator de regularização para tornar as splines mais suaves.

A proposta é um fator de suavização como alternativa. Esse fator controla o equilíbrio (trade-off) entre a capacidade do modelo e sua suavidade, exatamente o tipo de ajuste que esperamos de um bom mecanismo de regularização. A penalização sugerida pode ser descrita pela equação 3, em que $2m - 1$ corresponde ao grau da spline.

$$\frac{1}{2L}\int_{-L}^{L}\left(\phi^{(m)}(u)\right)^2, du. \tag{2}$$

A aplicação desse fator de regularização as *b-splines* que tradicionalmente compoem as KANs é algo até então inexplorado por conta da dificuldade de sua paralelização com GPUs.

3 Metodologia

Nesse projeto, a proposta é modelar convoluções KAN com splines, utilizando os pontos das splines como parâmetros para atualização com os algoritmos de atualização. A discussão busca abordar como o uso dessa técnica traz vantagens em termos de velocidade de treinamento e possibilita a introdução de outras ferramentas consolidadas de outras modalidades da modelagem estatística para esses modelos.

3.1 Descrição do modelo

A camada de convolução KAN estende o operador convolucional tradicional ao incorporar funções não-lineares nos elementos do kernel, baseados nas Kolmogorov-Arnold Networks. Para abordar alguns dos problemas observados na seção 2.3, optou-se por utilizar splines de primeiro grau em vez de b-splines. Essa modelagem traz algumas vantagens, possibilitando um treinamento mais rápido e novos mecanismos de regularização que serão discutidos em maior profundidade.

O modelo permite que cada filtro gere um canal de saída a partir da utilização dos canais de entrada para realizar interpolações lineares dos parâmetros, uma vez que cada função ϕ é uma spline de primeiro grau. A imagem 1 destaca como a aplicação de cada função é feita para gerar a imagem de saída.

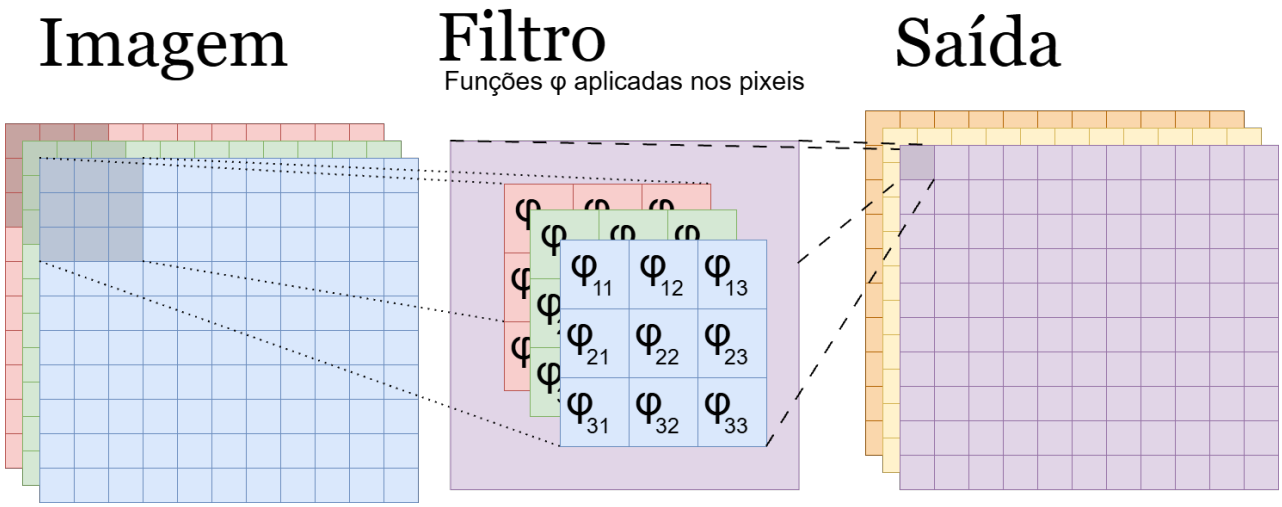

Figura 1: Diagrama de um filtro KANC aplicando as funções ϕ na convolução da imagem de entrada

Para a implementação de forma paralelizável com a GPU foram utilizadas operações vetoriais: Inicialmente, a imagem recebe zero-padding para preservar suas dimensões; em seguida, é decomposta em patches do tamanho do filtro, sobre os quais a operação será aplicada, como ilustrado pela imagem 2. Cada patch é então normalizado para um grid contínuo no intervalo $[0, 1]$, baseado no grid de entrada, onde estão definidos os pontos (igualmente espaçados) da spline. Dessa forma, a seleção de pontos vizinhos que são aplicados na interpolação é agilizada. Para cada posição dentro do patch, os pontos de suporte são escolhidos com base nesse valor normalizado, permitindo que o filtro realize uma interpolação linear

entre esses valores.

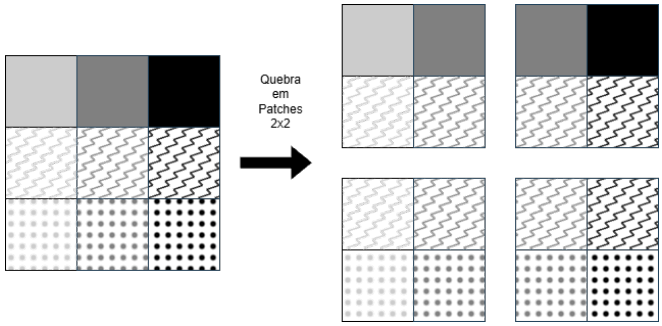

Figura 2: Diagrama da quebra da imagem 3x3 em patches 2x2

Por fim, as contribuições obtidas de todos os canais de entrada são somadas, produzindo a resposta correspondente a um filtro, e a agregação das respostas de todos os filtros compõe o volume final de saída, cujo número de canais é igual ao número total de filtros utilizados.

3.2 Regularizações

A abordagem de utilização dos parâmetros como pontos das splines de primeiro grau nos permite contornar o problema relatado em 2.3 de que as saídas costumam sair dos pontos definidos pelo *grid* de suporte da spline. Para minimizar esse vazamento nossa modelagem permite utilizar a regularização $L2$ para concentrar as saídas próximo do 0, uma vez que os pontos do grid são definidos no intervalo $[-1, 1]$. Pode-se observar na tabela 3 que conforme o fator de regularização aumenta o desvio padrão das saídas diminui e a média das ativações se concentra perto de 0.

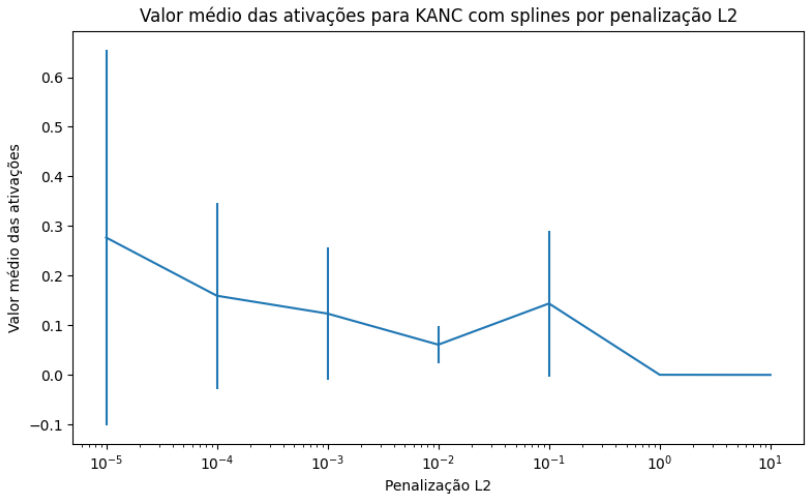

Figura 3: Gráfico das ativações internas da KANC por fator de penalização L2 em escala logarítmica

Além disso, nossa modelagem permite fazer uso da regularização descrita na seção 2.4, evitando funções excessivamente onduladas. Para nossas funções de primeiro grau, com pontos definidos de 1 a -1, temos que m = 1. Com isso, a regularização é descrita pela equação:

$$\frac{1}{2}\int_{-1}^{1}\left(\phi^{'}(u)\right)^{2},du. \tag{3}$$

Para ilustrar a forma como calcular a regularização de forma paralelizável, observe um exemplo de spline ϕ ilustrada na imagem 4. A derivada $\phi^{'}$ entre os pontos P_0 e P_1 é $P_1 - P_0$. A derivada dos demais intervalos é a diferença entre os pontos consecutivos. Somando a integral da derivada de todos os intervalos temos que o fator de regularização é a média do quadrado da diferença entre um ponto e o seu antecessor.Dessa maneira, é possível calcular esse valor para a camada de convolução com apenas algumas operações de tensores.

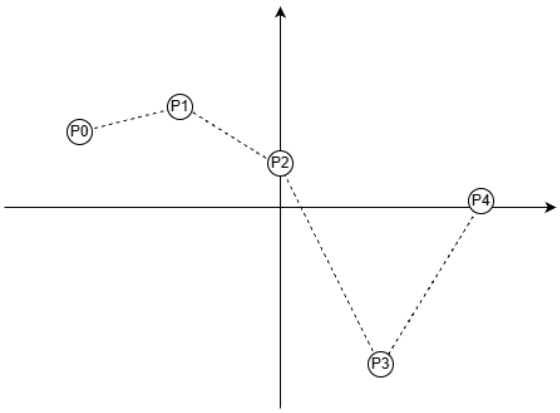

Figura 4: Função ϕ de exemplo, uma spline de 5 pontos

3.3 Inicialização dos parâmetros

Com o intuito de melhorar a convergência de uma camada sensível como a KANC, é necessário ter uma estratégia de inicialização de parâmetros. Para tanto, o objetivo da inicialização é garantir que a esperança e a variância da saída seja igual à da entrada.

No nosso caso, consideramos que a distribuição de um pixel da imagem de entrada é uma variável aleatória G uniforme no intervalo $(-1, 1)$ para cada um dos canais de entrada. Dessa maneira, cada valor tem esperança $E[G] = 0$ e variância $Var(G) = \frac{1}{3}$. Como pode-se observar pela figura 1, o valor da saída será a soma das funções ϕ aplicadas nos pixels.

Considerando a saída um filtro de altura H e largura L, com C canais, a saída do filtro é dada pela equação:

$$saída = \sum_{i=1}^{H} \sum_{j=1}^{L} \sum_{k=1}^{C} \phi_{ijk}(G)$$

Além disso, ajustando os pontos das funções ϕ igualmente espaçados no intervalo $(-1, 1)$, podemos descrever as funções ϕ ilustradas em 4 com funções degrau da seguinte forma:

$$\phi(G) = \sum_{m=1}^{n-1} deg_{[X_{m-1}, X_m]}(G) * interpolação_{[X_{m-1}, X_m]}(G)$$

E a função de interpolação γ :

$$\gamma_{[X_{m-1},X_m]}(G) = [Y_{m-1}(\frac{-1 + m(\frac{2}{n-1}) - G}{\frac{2}{n-1}}) + Y_m(\frac{1 - (m-1)(\frac{2}{n-1}) + G}{\frac{2}{n-1}})]$$

Na qual a spline tem n pontos, X_m é o valor de x em que se encontra o m-ésimo ponto da spline e Y_m seu valor, que será amostrado de uma distribuição uniforme Y que queremos encontrar. As funções degraus são definidas por:

$deg_{[X_{m-1},X_m]}(G) = 1$ caso $(X_{m-1} < G < X_m)$, e 0 caso contrário.

Dessa maneira queremos modelar a distribuição Y de forma que:

$Var(saída) = \frac{1}{3}$

E, calculando a variância da saída com as equações da KANC, temos:

$$Var(saída) = Var(\sum_{i=1}^{H}\sum_{j=1}^{L}\sum_{k=1}^{C}\phi_{ijk}(G))$$

As funções $\phi_{ij}(G)$ representam as funções de interpolação aplicadas em cada um dos pixeis da imagem. Pixeis próximos contém valores correlatos, mas para fins de estimação inicial dos parâmetros vamos considerar são independentes entre si e a distribuição é a mesma.

$$Var(saída) = HLC * Var(\phi_{ij}(G))$$

Como as interpolações são variáveis aleatórias independentes, a variância das funções $\phi_{ijk}(G)$ é:

$$Var(\phi(G)) = \sum_{m=1}^{n-1} Var(deg_{[X_{m-1},X_m]}(G) * \gamma_{[X_{m-1},X_m]}(G))$$

Cada um dos componentes somatórios da função é estritamente monótono e diferenciável em seu domínio, com isso conseguimos calcular a função de densidade de probabilidade f para cada $deg_{[X_{m-1},X_m]}(G) * \gamma_{[X_{m-1},X_m]}(G)$ é:

$$f_m(y) = \frac{1}{(n-1)(Y_{m-1} - Y_m)}$$

E, considerando todos os intervalos, a função de densidade de probabilidade é

$$f(y) = \sum_{m=1}^{n-1} \frac{1}{(n-1)(Y_{m-1} - Y_m)}$$

Calculando a esperança e a variância de uma função de densidade de probabilidade

contínua obtemos:

$$E[y] = \sum_{m=1}^{n-1} \int_{Y_{m-1}}^{Y_m} \frac{y * dy}{(n-1)(Y_{m-1} - Y_m)}$$

$$E[y] = \sum_{m=1}^{n-1} \frac{Y_m + Y_{m-1}}{2(n-1)}$$

Considerando que as variáveis Y_i são amostradas da mesma distribuição Y, temos que:

$$E[y] = E[Y]$$

E calculando a variância por:

$$E[y^2] = \sum_{m=1}^{n-1} \int_{Y_{m-1}}^{Y_m} \frac{y^2 * dy}{(n-1)(Y_{m-1} - Y_m)}$$

$$E[y^2] = \sum_{m=1}^{n-1} \frac{Y_m^2 + Y_m Y_{m-1} + Y_{m-1}^2}{3(n-1)}$$

Considerando novamente que as variáveis aleatórias são amostradas da mesma distribuição Y, temos que:

$$E^2[y] = \frac{1}{3}(2E[Y^2] + (E[Y])^2)$$

Agora, considerando que Y é uma distribuição uniforme, para que a esperança seja 0, como a da entrada da rede, o intervalo deve ser simétrico. Vamos defini-lá como uniforme contínua no intervalo $(-a, a)$, temos que:

$$E[Y] = 0$$

$$E[Y^2] = \frac{a^2}{3}$$

e

$$Var[y] = E[y^2] - (E[y])^2 = \frac{1}{3}(2E[Y^2] + (E[Y])^2) - 0^2$$

$$Var(\phi) = Var[y] = \frac{2a^2}{9}$$

Agora, substituindo para encontrar a variância da camada obtemos:

$$Var(saída) = HLC * \frac{2a^2}{9}$$

Como queremos modelar para que a variância da saída seja $\frac{1}{3}$, isolamos a:

$$a = \sqrt{\frac{3}{2HLC}}$$

Em nossa modelagem, portanto, inicializaremos os parâmetros seguindo a distribuição $Y \sim Uniforme(-\sqrt{\frac{3}{2HLC}}, \sqrt{\frac{3}{2HLC}})$

Vale ressaltar que essa análise foi feita para a primeira camada convolucional Kolmogorov-Arnold. Para as demais, a distribuição de entrada não será uma distribuição uniforme, e sim uma soma da interpolação de uniformes. Contudo, essa inicialização também é uma boa estimativa para esses casos, pois, embora não sejam uniformemente distribuídos, o resultado ainda é a soma da interpolação para entradas de mesma esperança e variância.

3.4 Comparações de velocidade dos modelos

Utilizando uma máquina com uma GPU L4, o dataset FashionMNIST [8] e a arquitetura igual àquela descrita na subseção 2.3 conseguimos comparar a velocidade de execução dos dois modelos. Os modelos escolhidos para a comparação dos tempos de execução foram os KANC MLP, descritos na imagem 5. Os tempos de treinamento foram coletados utilizando o dataset inteiro em batches de 64 como na referência. Para assegurar a confiabilidade do resultado foi executada a média de 10 experimentos para obter o tempo final de treinamento por época.

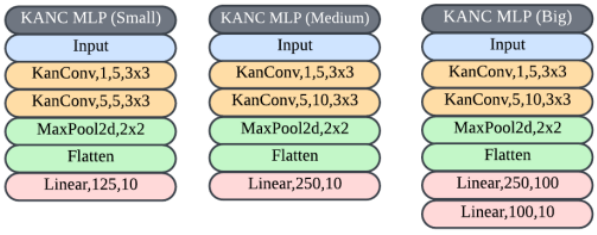

Figura 5: KANC MLPs do artigo CONVOLUTIONAL KOLMOGOROV-ARNOLD NETWORKS

3.5 Otimização local no espectro

Uma das propostas exploradas com essa abordagem de uso das splines para modelagem das funções ϕ é a capacidade dessas redes de realizar a otimização local no espectro da imagem. Como para cada entrada da camada de convolução apenas dois dos K parametros da spline são ativados e otimizados, é possível otimizar a detecção de features para uma parte do espectro, especializando as redes para as últimas amostras de treinamento sem afetar seu desempenho com as demais amostras.

Para comprovar essa capacidade, foi feita uma comparação entre a média da diferença das representações internas entre as duas camadas convolucionais discutidas após o treinamento em uma amostra. Utilizou-se a arquitetura da figura 7, e utiliza-se como ativação interna a saída da última camada convolucional. O treinamento é feito apenas em uma amostra e a diferença exibida no gráfico da figura 6 é a razão da média da diferença no dataset de validação valor antes e depois da atualização do treinamento pelo valor da diferença para essa amostra treinada. Uma rede com aprendizado local busca maximizar a diferença para a amostra de treinamento sem alterar o valor para as demais amostras, minimizando o valor da razão. Observe que conforme o número de pontos da spline de convolução aumenta, essa razão diminui, acentuando o fato de que com mais pontos o aprendizado da KANC é cada vez mais local.

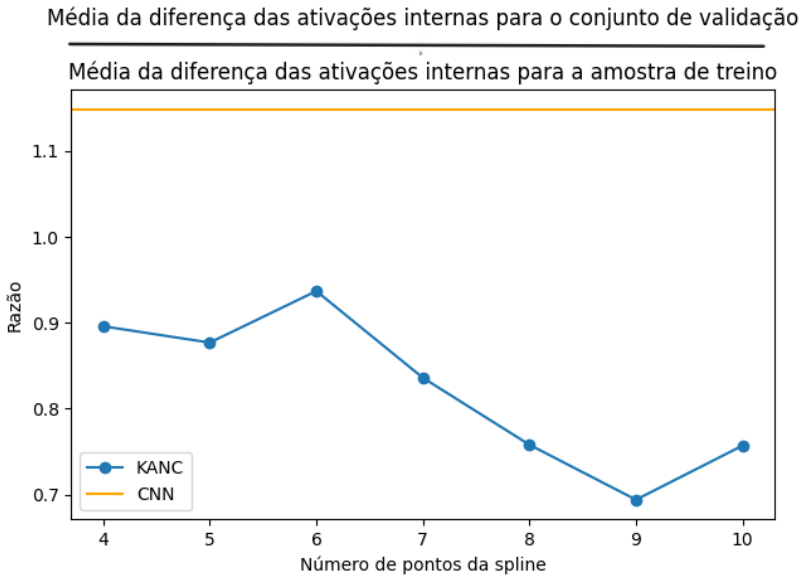

Figura 6: Razão entre média do valor das ativações internas para o conjunto de validação sobre a média de atualização para a amostra de treino para as redes CNN e KANC MLP, por número de pontos da spline ϕ

Outro experimento que busca explorar a capacidade de otimização local foram os testes de treinamento utilizando o dataset CIFAR10, com técnicas de aumentação.

3.5.1 Treinamento das redes

Para realizar o treinamento das redes foi feita a implementação da arquitetura em PyTorch e o carregamento do dataset CIFAR10 utilizando *helpers* da própria biblioteca. A máquina utilizada tem 12 GB de memória RAM utilizando uma GPU L4.

Antes do treinamento das redes separou-se 20% das amostras de treino para validação. Embora o treinamento da maior parte dos algoritmos de aprendizado profundo utilize o otimizador Adam, no nosso caso suas aplicações são limitadas.

O Adam é definido pela agregação de duas técnicas de otimização conhecidas, junto com a descida do gradiente: O momentum e o RMSProp. Como no caso da modelagem por splines as derivadas das funções não são monotônicas, o uso do momentum levando em consideração uma média móvel das atualizações antigas pode dificultar a convergência do treinamento dessa rede sensível. Por conta disso, optou-se por fazer o treinamento dessa rede em cima do algoritmo de otimização RMSProp, que embora não publicado pode ser

encontrado em [7].

As redes treinadas são CNNs e KANCs, para comparação das duas arquiteturas. Para ambas arquiteturas são aplicadas 3 camadas de convolução seguidas de uma camada MLP de classificação. Com o intuito de comparar a qualidade das convoluções foi feita a escolha de não se utilizar mais camadas densas. Nos dois casos os filtros são 3x3 e no caso das KANCs foi feita a escolha de utilizar o grid de pontos entre -1 e 1. Contudo, o número de filtros foi considerado para ambos os casos um hiperparâmetro, assim como o número de pontos no caso das KANCs.

Além disso, para comprovar a hipótese de que a aplicação da camada KANC tem grande potencial de extração de *features* visuais para modelos ViT, esse modelo foi abordado no projeto, com a VIT CNN que usa uma camada de convolução típica e o KANC que usa a convolução Kolmogorov-Arnold. Considerando o uso tradicional de uma camada convolucional antes da geração das palavras para as camadas de atenção, os filtros da camada KANC conseguem gerar representações muito mais concisas em termos de espaço, possibilitando um menor desperdício de parâmetros nas camadas downstream do modelo.

Adicionalmente, quando acoplada à uma rede ViT, essa camada não tem seu maior defeito evidenciado: A composição de funções não-suaves que provavelmente dificulta seu treinamento. Isso não aconteceria em uma rede com apenas uma camada KANC, como a literatura acerca de ViTs sugere com uma camada convolucional.

Os modelos estão descritos na imagem 7

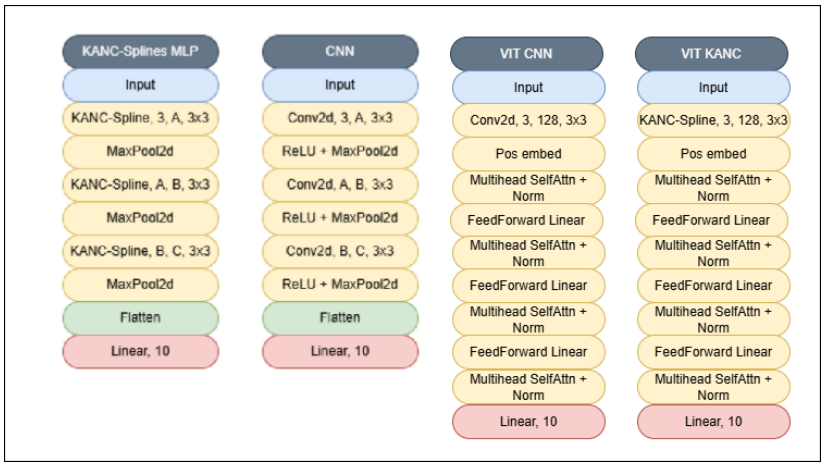

Figura 7: Arquiteturas KANC MLP e CNN utilizadas. A, B e C são hiperparâmetros. Arquiteturas VITCNN e VITKANC

3.5.2 Técnicas de aumentação

Para comprovar a capacidade de otimização da extração de features específica no espectro foram aplicadas técnicas de aumentação para as imagens de treino, validação e teste. Foram aplicadas as técnicas de ColorJitter do PyTorch, alterando aleatóriamente as propriedades de luminosidade, saturação, matiz e contraste da imagem.

A aumentação é parametrizada nos experimentos por um fator de aumentação, quanto maior esse valor mais agressiva a transformação na imagem. Os valores dos parâmetros de ColorJitter parar um fator de aumentação λ são compostos da seguinte forma.

distorção de luminosidade = λ

distorção de contraste = λ

distorção de saturação = $\lambda \times 0.6$

distorção de matiz = $\lambda \times 0.1$

3.5.3 Busca de hiperparâmetros

Em ambos os modelos temos hiperparâmetros que afetam o treinamento e a qualidade do modelo resultante, para encontrar bons valores foi feita uma busca de hiperparâmetros utilizando o framework optuna [1]. A biblioteca conta com o Estimador Parzen estruturado em árvore (TPE) que utiliza os valores objetivos dos treinos passados para sugerir um novo conjunto promissor de hiperparâmetros. Além disso, o framework também conta com ferramentas de *Pruning*, interrompendo o treinamento de conjuntos não-promissores. O valor objetivo para o treino foi o erro médio da entropia cruzada do conjunto de validação.

3.5.4 Métricas utilizadas

A métrica utilizada para a busca de hiperparâmetros foi o erro médio de entropia cruzada no conjunto de validação, mas depois de encontrados esses hiperparâmetros nós fixamos os *magic-numbers* de randomização a fim de realizar experimentos reprodutíveis. Nesses experimentos avaliamos o desempenho dos modelos finais utilizando algumas métricas de base de comparação:

Erro de Entropia Cruzada: O erro de entropia cruzada, também conhecido como *cross-entropy loss*, é amplamente utilizado em problemas de classificação, especialmente em aprendizado profundo. Para um conjunto de dados com N amostras e C classes, a entropia cruzada é definida como:

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log \hat{y}_{ic},$$

onde y_{ic} é o rótulo verdadeiro one-hot codificado para a classe c da amostra i, e $\hat{y}_{ic}$ é a probabilidade prevista pelo modelo para essa mesma classe. Essa métrica penaliza previsões à classes incorretas, ensinando a classificação ao modelo.

Precisão: A precisão quantifica a proporção de amostras de uma classe (verdadeiros positivos) dentre aquelas classificadas como pertencentes a essa classe pelo modelo.

$$\text{Precisão} = \frac{TP}{TP + FP},$$

onde TP denota verdadeiros positivos e FP falsos positivos.

Recall: O *recall* mede a capacidade do modelo de identificar corretamente todas as amostras daquela classe (amostras positivas):

$$\text{Recall} = \frac{TP}{TP + FN},$$

onde FN representa falsos negativos.

F1-Score: O F1-score representa a média harmônica entre precisão e recall, fornecendo um equilíbrio entre ambas as métricas:

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}.$$

O F1-score é particularmente útil quando existe um desbalanceamento entre classes, pois combina as duas métricas de forma a refletir melhor o desempenho global do modelo.

3.5.5 Métodos de ensemble

Além da comparação dos modelos CNN e KANC, também conseguimos avaliar a qualidade do ensemble desses dois modelos. A arquitetura de cada um dos modelos reforça qualidades complementares: A otimização global e eficiente das CNNs enquanto os KANCs são otimizadores locais do espectro. Com isso, como modelos de características distintas conseguem se complementar com bons desempenhos no ensemble, vamos comparar o desempenho dos dois modelos com um ensemble com a técnica de média dos logitos. Foi feito o ensemble da rede convolucional (CNN) com a rede convolucional Kolmogorov-Arnold (KANCs) de forma a tirar a média da classificação das duas redes gerando uma nova classificação, que também foi avaliada acerca das métricas.

4 Resultados

Nessa seção vamos discutir os resultados obtidos dos experimentos descritos na metodologia.

4.1 Comparação dos tempos de treinamento

Utilizando a metodologia previamente descrita em 3.4, treinamos no FashionMNIST os modelos KANC MLP small, medium and large comparando o tempo de treinamento entre a técnica da referência 2.3 e nosso modelo descrito em 3.1 pela tabela 4.1. É possível observar um grande ganho na velocidade de treinamento desses modelos, obtendo um treinamento entre 6 e 14 vezes mais rápido para esses casos.

Tabela 1: Tempo em minutos de treinamento por época no dataset FashionMNIST

Configuração	**B-Splines**	**Splines (nosso)**
small (gs 10)	1.17	**0.17**
small (gs 20)	1.63	**0.17**
medium (gs 10)	1.64	**0.17**
medium (gs 20)	2.46	**0.17**
big (gs 10)	1.65	**0.17**
big (gs 20)	2.46	**0.18**

4.2 Desempenho de classificação para CIFAR10

Na avaliação realizada no dataset CIFAR10, de acordo com o descrito em 3.5.1, temos a performance dos diferentes modelos nas métricas da seção 3.5.4. Os modelos avaliados foram o convolucional (CNN), o modelo convolucional Kolmogorov-Arnold com uma camada MLP (KANC MLP) e, por fim, esse modelo treinado com as regularizações descritas em 3.2, com o foco na reprodutibilidade dos resultados, originando a tabela 4.2.

Como pode-se observar, os modelos KANC MLP com (identificado por _reg) e sem regularização tiveram métricas inferiores nas três dimensões analizadas, refletindo uma performance inferior na tarefa de classificação do dataset em comparação com as redes CNNs. Contudo, também é possível observar na tabela como a performance das redes CNNs decai de acordo com o fator de aumentação, o que não ocorre para as redes KANC MLPs, destacando sua capacidade de otimização local no espectro. Para essas redes o fator de aumentação de 0.1 apresentou maiores dificuldades para as redes, que tiveram melhores performances em fatores de aumentação maiores.

Tabela 2: Performance dos modelos no dataset CIFAR10 com níveis variáveis de aumentação

Model	Precision	Recall	F1-score
Sem aumentação			
CNN	0.703	0.706	0.702
KANC MLP	0.635	0.633	0.632
KANC_reg MLP	0.665	0.605	0.598
CNN + KANC_reg MLP	**0.730**	**0.721**	**0.720**
ViT CNN	0.674	0.668	0.668
ViT KANC	0.595	0.591	0.588
Com fator de aumentação = 0.1			
CNN	0.632	0.620	0.622
KANC MLP	0.530	0.531	0.520
KANC_reg MLP	0.572	0.572	0.568
CNN + KANC_reg MLP	**0.654**	**0.649**	**0.649**
ViT CNN	0.524	0.526	0.523
ViT KANC	0.529	0.528	0.520
Com fator de aumentação = 0.3			
CNN	0.622	0.616	0.616
KANC MLP	0.571	0.565	0.561
KANC_reg MLP	0.605	0.607	0.604
CNN + KANC_reg MLP	**0.659**	**0.659**	**0.658**
ViT CNN	0.529	0.520	0.521
ViT KANC	0.542	0.539	0.538
Com fator de aumentação = 0.5			
CNN	0.618	0.612	0.612
KANC MLP	0.569	0.564	0.558
KANC_reg MLP	0.606	0.607	0.606
CNN + KANC_reg MLP	**0.655**	**0.653**	**0.652**
ViT CNN	0.535	0.522	0.524
ViT KANC	0.558	0.556	0.550

Também se destaca a performance dos ensembles de CNNs com KANC MLPs regularizados, apresentando o melhor desempenho nas métricas escolhidas dentre todos os fatores de aumentação. Outra propriedade notável desse arranjo é como a performance melhorou, e, simultaneamente, a capacidade das KANC MLPs de não abaixar o desempenho de acordo com o aumento do fator de aumentação. Embora a hipótese de que as redes KANC MLPs se adaptassem melhor à mudança de espectro, seu desempenho não foi superior ao desempenho das redes CNNs.

Adicionalmente, comparando o desempenho das KANCs com regularização e sem é

possível observar que em todos os cenários com aumentação a regularização trouxe ganhos no resultado, e no cenário sem aumentação se observa uma precisão maior. Com isso, podemos observar que as regularizações melhoram o desempenho e a generalização dessas redes em tarefas de classificação.

Comparando os resultados para os modelos de arquitetura *vision transformer* nota-se que os modelos com a camada convolucionarl KANC obtiveram um desempenho mais consistente conforme as variações no fator de aumentação. O desempenho da rede com a camada convolucional Kolmogorov-Arnold supera o com a camada convolucional padrão para fatores de aumentação diferente de 0. Isso sugere que essa nova camada pode melhorar o desempenho de modelos ViT em cenários de alta variação de luminosidade, progredindo em modelos estado-da-arte.

Os experimentos podem ser encontrados nesse link do drive.

5 Conclusões

Neste trabalho, apresentamos uma maneira de realizar convoluções com funções splines no lugar de kernels de multiplicação de fatores, com inspiração nas redes Kolmogorov-Arnold. Mostramos como uma modelagem utilizando os pontos das splines como parâmetros traz vantagens em relação à abordagem tradicional por b-splines. Também conseguimos observar a capacidade das redes que utilizam camadas KANCs de se adaptarem a cenários de alteração no espectro das imagens, destacando o potencial de aprendizagem local dessa camada.

6 Limitações e Trabalhos Futuros

Nossa estratégia de quebrar a imagem em patches para aplicação dos filtros de convolução usa grandes quantidades de memória das GPUs, o que sugere que essa abordagem não é a ideal para imagens grandes. Contudo, uma abordagem utilizando computação de estêncil pode ser capaz de levar essa abordagem à aplicações reais de classificação de imagem. Para comprovar o desempennho dessa camada em modelos ViT também será necessário um processo de treinamento mais longo e uma busca de hiperparâmetros mais profunda. Outra limitação dessa camada em modelos ViT é que a regularização L2 dos parâmetros da KAN é diferente daquela para os demais parâmetros da rede. Como as camadas **feed-forward** tem seus parâmetros como multiplicadores da entrada e não interpoladores, a regularização L2 junta reduz o nível de representação da camada convolucional, sendo necessários dois otimizadores (um com cada regularização L2).

Além desse hiperparâmetro adicional de regularização L2 dos parâmetros, introduzimos outro fator de regularização na derivada das splines e o número de pontos da spline. A adição desses hiperparâmetros dificulta o processo de **grid-search**, volatilizando o treinamento desses modelos.

Referências

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[2] Alexander Dylan Bodner, Antonio Santiago Tepsich, Jack Natan Spolski, and Santiago Pourteau. Convolutional kolmogorov-arnold networks, 2025.

[3] Peter Craven and Grace Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31(4):377–403, 1978.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[6] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov–arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[7] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[8] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.