Erros de classificação e test time augmentation

Carlos Eduardo Fontaneli RA 769949
Leticia email@domain
Luís Augusto Simas RA 790828

1 Resumo

A robustez e precisão dos modelos de classificação de imagens representam desafios no campo da visão computacional. A técnica de "Test Time Augmentation" (TTA) surge como uma metodologia para enfrentar tais desafios, particularmente no contexto de erros de classificação. Este estudo, desenvolvido no âmbito da disciplina de Visão Computacional ofertada pela UFSCar em 2024, foca na aplicação de TTA para mitigar erros de classificação utilizando uma ResNet-18 no dataset Oxford Pets.

2 Introdução

A complexidade dos modelos de aprendizado de máquina e as limitações nos conjuntos de dados de treinamento resultam em desafios significativos para alcançar desempenhos satisfatórios durante a fase de teste. Em resposta a essas limitações, a técnica de Test-Time Augmentation(TTA), detalhada em Kimura (2024), surge como uma estratégia para melhorar a acurácia e a robustez dos modelos em aplicações de visão computacional. Este método aproveita a abordagem de data augmentation durante a fase de teste, não apenas para melhorar a representação dos dados diversificando-os, mas também para reduzir a discrepância entre o desempenho durante o treino e o teste.

A principal motivação para o uso do TTA reside na sua capacidade de simular um ambiente de ensemble learning(também definido como aprendizado em conjunto) sem a necessidade de treinar múltiplos modelos. Ao aplicar transformações variadas nas imagens durante a fase de teste e tomar a média das saídas do modelo, o TTA permite uma estimativa mais confiável da classe verdadeira de cada imagem. Este processo não apenas aumenta a diversidade dos dados visualizados pelo modelo, mas também ajuda a mitigar os efeitos de overfitting em relação ao conjunto de treino, fornecendo uma visão mais geral e abrangente das representações dos dados de entrada.

Em visão computacional, as imagens capturadas em condições reais podem variar significativamente devido a alterações na iluminação, orientação e fundo. Essas variações podem confundir modelos que foram treinados em um conjunto de dados relativamente homogêneo ou insuficiente. O TTA aborda esse problema ao testar o modelo com várias versões da mesma imagem, garantindo que o modelo seja robusto a variações menores.

Embora o TTA tenha demonstrado ser eficaz em experimentos práticos, uma discussão teórica abrangente sobre seus fundamentos ainda está em desenvolvimento. Este projeto não apenas explora a aplicabilidade do TTA em um cenário prático específico, mas também busca contribuir para a compreensão teórica da técnica. Através da análise das situações em que o modelo original falhou, esperamos elucidar como o TTA melhora a generalização dos modelos de aprendizado de máquina.

No entanto, é importante destacar que o uso de TTA implica um aumento considerável no custo computacional, uma vez que múltiplas versões de cada imagem precisam ser processadas pelo modelo. Consequentemente, essa técnica pode apresentar limitações em termos de escalabilidade e eficiência em ambientes de produção onde o tempo de inferência é crítico. Ademais, mesmo quando a test-time augmentation resulta em melhoria da acurácia, a técnica pode alterar previsões corretas para incorretas, demonstradas em Shanmugam et al. (2020)

3 Objetivos

O objetivo deste projeto é explorar a eficácia da técnica de Test Time Augmentation (TTA) na melhoria das predições de um modelo de classificação de imagens, utilizando uma ResNet-18 treinada para classificar o dataset Oxford Pets.

Em seguida, o projeto visa identificar as 10 ima-

gens de cachorro e as 10 imagens de gato do conjunto de validação para as quais o modelo apresentou as piores predições.

Além disso, objetiva-se analisar as mudanças nas classificações dessas 20 imagens após a aplicação da técnica de TTA. Serão avaliadas as principais melhorias e pioras nas predições. Diferentes pipelines de augmentation serão testados para identificar quais proporcionam os melhores e os piores resultados.

Ademais, visando o aprofundamento da análise do TTA, aplicou-se a mesma sequência de passos para as 10 melhores predições do modelo, em busca de confirmar se haverá piora de resultado.

4 Metodologia

A técnica de Test Time Augmentation (TTA) baseia-se em uma heurística fortemente influenciada pelas técnicas de data augmentation e ensemble learning. A técnica se baseia na ideia de utilizar data augmentation durante o processo de inferência e fazer uma agregação dos resultados do modelo aplicado aos dados aumentados obtidos.

Como apresentado em Kimura (2024), a TTA ainda carece de discussões sistemáticas com relação a seus aspectos teóricos. Apesar disso, sua eficácia em experimentos práticos pode ser observada, principalmente quando aplicada à tarefas de visão computacional como classificação de imagens.

4.1 Definição do método

Seja x a entrada original para o modelo e $f(x)$ a função de predição do modelo. Tomamos um conjunto de transformações $T = \{t_1, t_2, \ldots, t_n\}$ no qual t_i representa uma função de transformação (rotação, mudança de escala, jitter etc).

Sendo assim, a técnica de TTA consiste em aplicar cada transformação em T à entrada original x, obtendo um conjunto X' de entradas, e então aplicar o modelo à cada uma das entradas em X', agregando as predições obtidas em uma predição final $\hat{y}$.

Dado um modelo f, uma entrada x e um conjunto de transformações $T = \{t_1, t_2, \ldots, t_n\}$ a predição final $\hat{y}$ é definida como:

$$\hat{y} = A(\{f(t_i(x))\}_{i=1}^{n})$$

onde A é dita a *função de agregação* da TTA.

4.2 Funções de agregação

O resultado da técnica de TTA depende não somente das transformações utilizadas, mas também da função de agregação A a ser aplicada ao conjunto X' de predições obtidas. Tendo isso em vista, o processo de escolha da função de agregação pode ser influenciado pelo problema alvo do modelo e também pelo conhecimento prévio das funções de transformação e as invariâncias do modelo com relação a tais transformações.

Em geral, os métodos de agregação mais comuns utilizados na técnica de TTA são as médias simples e ponderadas. Agregar os resultados utilizando uma média ponderada se torna especialmente relevante quando as transformações têm propriedades previamente conhecidas e deseja-se dar maior peso à resultados obtidos por transformações específicas. Ademais, em Kimura (2024) também é proposta uma técnica para derivar pesos ótimos.

Técnicas de agregação temporal também podem ser utilizadas para tarefas de detecção de anomalias em redes Cohen et al. (2023).

4.3 Implementação da técnica de TTA

A função `test_time_augmentation` aplica a `augmentation_pipeline` `K` vezes a cada imagem presente em `images`. A ideia é, para cada imagem em um batch, gerar um novo batch de `K` imagens através da aplicação da `augmentation_pipeline` à imagem.

Nesse caso, considera-se que a `augmentation_pipeline` aplica transformações à imagem original de maneira não determinística. Dessa forma, aplicando a pipeline `K` vezes à uma dada imagem, obtém-se um conjunto de `K` novas imagens.

```
def test_time_augmentation(
    images, augmentation_pipeline, K
):
    batch_augmented_images = []

    for image in images:
        augmented_images = []
        for _ in range(K):
            augmented_image = (
                augmentation_pipeline(
                    image
                )
            )
            augmented_images.append(
                augmented_image
```

```
            )

        batch_augmented_images.append(
            augmented_images
        )

    return batch_augmented_images
```

Com isso, é possível gerar um conjunto X' com K imagens para cada imagem x para a qual se deseja aplicar a técnica de TTA. Sendo assim, para obter a predição desejada para a imagem x, basta aplicar o modelo às K imagens obtidas no passo anterior e então aplicar uma função de agregação às predições obtidas. Nesse caso a função de agregação adotada é a média simples dos valores da predição.

```
def make_predictions_with_tta(
    model,
    batch_augmented_images,
    device=AVAILABLE_DEVICE,
):
    softmax = nn.Softmax(dim=1)
    all_avg_probs = []

    with torch.no_grad():
        for (
            augmented_images
        ) in batch_augmented_images:
            all_probs = []
            for (
                image
            ) in augmented_images:
                image = image.to(
                    device
                ).unsqueeze(
                    0
                )
                output = model(image)
                all_probs.append(
                    output.cpu()
                )

            avg_prob = torch.mean(
                torch.stack(all_probs),
                dim=0,
            )
            all_avg_probs.append(
                softmax(
                    avg_prob
                ).squeeze(0)
            )

    return all_avg_probs
```

4.4 Pipelines testados

Neste projeto, foram testados cinco pipelines de augmentation diferentes para avaliar a eficácia da técnica de Test Time Augmentation (TTA). Cada pipeline aplica uma série de transformações às imagens originais, gerando versões aumentadas que são usadas durante a inferência. Abaixo, estão descritos os pipelines utilizados:

- **Pipeline 1**: Este pipeline aplica um flip horizontal aleatório com 50% de probabilidade, seguido por uma rotação aleatória de até 15 graus. Além disso, ajustes aleatórios de brilho, contraste, saturação e matiz são realizados, e um recorte redimensionado aleatório é aplicado à imagem.

```
augmentation_pipeline_1 = T.Compose([
    T.RandomHorizontalFlip(p=0.5),
    T.RandomRotation(degrees=15),
    T.ColorJitter(brightness=0.2,
    contrast=0.2,
    saturation=0.2,
    hue=0.1),
    T.RandomResizedCrop(
    size=(224, 224),
    scale=(0.8, 1.0))
])
```

- **Pipeline 2**: Este pipeline utiliza um flip vertical aleatório com 50% de probabilidade, seguido por uma rotação aleatória de até 30 graus. Também são aplicados ajustes mais intensos de brilho, contraste, saturação e matiz, juntamente com transformações de afinidade, como tradução, escala e cisalhamento, e, por fim, a imagem é redimensionada para 224x224.

```
augmentation_pipeline_2 = T.Compose([
    T.RandomVerticalFlip(p=0.5),
    T.RandomRotation(degrees=30),
    T.ColorJitter(brightness=0.3,
    contrast=0.3,
    saturation=0.3,
    hue=0.2),
    T.RandomAffine(degrees=0,
    translate=(0.1, 0.1),
    scale=(0.9, 1.1),
    shear=10),
```

```
    T.Resize(size=(224, 224))
])
```

- **Pipeline 3**: Com uma probabilidade de 70%, este pipeline aplica um flip horizontal aleatório seguido por uma rotação de até 45 graus. São realizados ajustes leves de brilho, contraste, saturação e matiz, uma transformação de perspectiva aleatória, e um recorte redimensionado aleatório com uma escala diferente.

```
augmentation_pipeline_3 = T.Compose([
    T.RandomHorizontalFlip(p=0.7),
    T.RandomRotation(degrees=45),
    T.ColorJitter(brightness=0.1,
    contrast=0.1,
    saturation=0.1,
    hue=0.05),
    T.RandomPerspective(
    distortion_scale=0.5,
    p=0.5),
    T.RandomResizedCrop(
    size=(224, 224),
    scale=(0.7, 1.0))
])
```

- **Pipeline 4**: Este pipeline combina flips horizontais e verticais aleatórios, cada um com 50% de probabilidade. Aplica uma rotação de até 90 graus, ajustes mais intensos de brilho, contraste, saturação e matiz, conversão para escala de cinza com 20% de probabilidade, e um recorte redimensionado aleatório.

```
augmentation_pipeline_4 = T.Compose([
    T.RandomHorizontalFlip(p=0.5),
    T.RandomVerticalFlip(p=0.5),
    T.RandomRotation(degrees=90),
    T.ColorJitter(brightness=0.4,
    contrast=0.4,
    saturation=0.4,
    hue=0.3),
    T.RandomGrayscale(p=0.2),
    T.RandomResizedCrop(
    size=(224, 224),
    scale=(0.6, 1.0))
])
```

- **Pipeline 5**: Este pipeline aplica um flip horizontal aleatório com alta probabilidade (80%), seguido por uma rotação suave de até 10 graus. Também realiza ajustes muito intensos de brilho, contraste, saturação e matiz, inverte as cores da imagem com 30% de probabilidade, e aplica um recorte redimensionado aleatório com uma escala variada.

```
augmentation_pipeline_5 = T.Compose([
    T.RandomHorizontalFlip(p=0.8),
    T.RandomRotation(degrees=10),
    T.ColorJitter(brightness=0.5,
    contrast=0.5,
    saturation=0.5,
    hue=0.4),
    T.RandomInvert(p=0.3),
    T.RandomResizedCrop(
    size=(224, 224),
    scale=(0.5, 1.0))
])
```

Esses pipelines foram aplicados nas imagens para gerar versões aumentadas, as quais foram posteriormente utilizadas para realizar a inferência e avaliar a eficácia do TTA em melhorar as predições do modelo.

4.5 Análise do custo computacional

Apesar dos potenciais ganhos em acurácia e robustez trazidas pelo TTA, é importante destacar as implicações de performance da aplicação dessa técnica em ambientes produtivos. Como a técnica consiste em gerar n novas imagens para cada predição, o custo adicional da técnica consiste não somente no aumento da quantidade de predições a ser realizada, mas também no custo de gerar as novas imagens.

Seja C o custo de realizar a predição para uma imagem x e G o custo de gerar uma nova imagem a partir da imagem x. Dada a aplicação da técnica de TTA com a geração de n novas imagens e uma função de agregação com custo A, o custo computacional final C' da predição realizando a técnica de TTA é dado por:

$$C' = n \cdot (C + G) + A$$

É importante notar que, para se obter uma estimativa acurada do custo computacional da aplicação da técnica de TTA em um ambiente real, outros fatores devem ser considerados como a capacidade de batch do hardware utilizado e potenciais otimizações para melhor explorar tal capacidade.

5 Resultados

Nesta seção, são apresentados os resultados da aplicação da técnica de Test Time Augmentation (TTA).

5.1 Resultados para Imagens com as Piores Predições

Antes da aplicação do TTA, o modelo apresentou um desempenho significativamente baixo, com uma acurácia de 20% e um F1-score de 0.0, evidenciando uma incapacidade quase total de classificar corretamente as imagens. No entanto, após a aplicação do TTA, houve uma melhoria substancial na melhor iteração, onde a acurácia atingiu 55% e o F1-score subiu para 0.47, demonstrando a eficácia do TTA em melhorar as predições.

Mesmo na pior iteração do TTA, o desempenho do modelo foi superior ao estado inicial, com uma acurácia de 45% e F1-score de 0.27. A média das iterações do TTA resultou em uma acurácia de 50% e F1-score de 0.37, indicando uma melhora consistente em relação ao desempenho sem TTA. Como é possível ver na figura1:

```
Metrics Before TTA:
Accuracy: 0.2000
Precision: 0.0000
Recall: 0.0000
F1 Score: 0.0000

Metrics After TTA (Best Iteration):
Accuracy: 0.5500
Precision: 0.5714
Recall: 0.4000
F1 Score: 0.4706

Metrics After TTA (Worst Iteration):
Accuracy: 0.4500
Precision: 0.4000
Recall: 0.2000
F1 Score: 0.2667

Metrics After TTA (Mean of Iterations):
Accuracy: 0.5000
Precision: 0.4943
Recall: 0.3000
F1 Score: 0.3685
```

Figure 1: Resultados das métricas para o caso das piores predições.

A análise da matriz de confusão, disponível na figura1, reforça esses achados. Antes do TTA, o modelo apresentou dificuldades, com apenas 4 imagens de gatos sendo corretamente classificadas, e todos os cães foram incorretamente classificados como gatos. Após a aplicação do TTA, a melhor iteração corrigiu 3 das 6 classificações incorretas de gatos e conseguiu classificar corretamente 4 dos 10 cães. No entanto, a pior iteração e a média das iterações mostraram que o modelo ainda enfrenta desafios significativos na classificação correta dos cães, com um número considerável de erros persistindo.

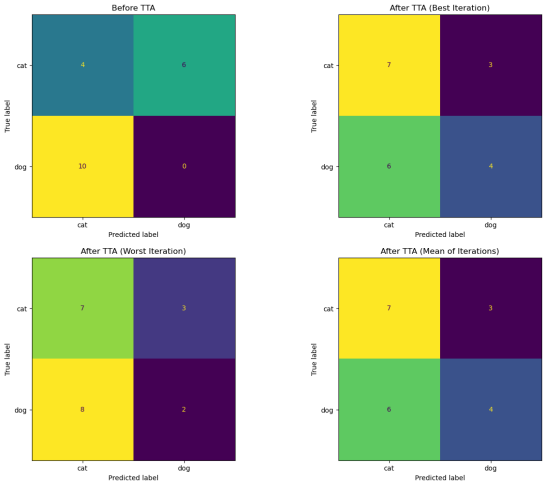

Figure 2: Resultados das métricas para o caso das piores predições.

5.2 Resultados para Imagens com as Melhores Predições

Para as imagens que o modelo classificou corretamente antes da aplicação do TTA, o desempenho inicial foi perfeito, com acurácia e F1-score de 1.0. Após a aplicação do TTA, observou-se uma leve queda de desempenho na melhor iteração, com a acurácia diminuindo para 95% e o F1-score caindo para 0.95, ainda assim mantendo um desempenho excelente.

A pior iteração do TTA revelou uma queda mais acentuada, com acurácia de 80% e F1-score de 0.83, sugerindo que o TTA pode, em alguns casos, introduzir incertezas que prejudicam as predições. A média das iterações resultou em uma acurácia de 88% e F1-score de 0.89, indicando que, embora o TTA continue oferecendo um desempenho robusto, ele ainda está aquém do desempenho perfeito inicial. Como é possível ver na figura3:

A matriz de confusão, disponível na figura4, confirma essa análise. Antes do TTA, o modelo conseguiu classificar corretamente todas as imagens de gatos e cães. Após o TTA, especialmente na pior iteração, o modelo cometeu 4 erros de classificação na classe "cat". A melhor iteração resultou

```
Metrics Before TTA:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000

Metrics After TTA (Best Iteration):
Accuracy: 0.9500
Precision: 0.9091
Recall: 1.0000
F1 Score: 0.9524

Metrics After TTA (Worst Iteration):
Accuracy: 0.8000
Precision: 0.7143
Recall: 1.0000
F1 Score: 0.8333

Metrics After TTA (Mean of Iterations):
Accuracy: 0.8800
Precision: 0.8491
Recall: 0.9400
F1 Score: 0.8874
```

Figure 3: Resultados das métricas para o caso das piores predições.

em apenas um erro para "cat", enquanto a média das iterações apresentou 2 erros. Isso sugere que, embora o TTA possa ser benéfico em alguns casos, ele também tem o potencial de introduzir erros em um modelo previamente perfeito.

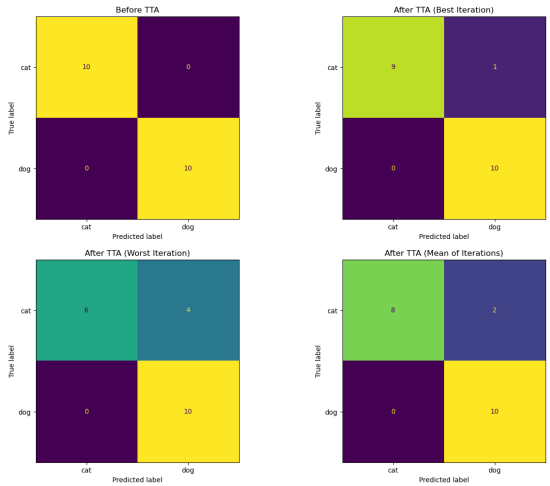

Figure 4: Resultados das métricas para o caso das piores predições.

6 Conclusão

Os resultados demonstram que a técnica de Test Time Augmentation (TTA) pode melhorar significativamente as predições em casos onde o modelo inicialmente falha, aumentando tanto a acurácia quanto o F1-score. No entanto, também foi observado que o TTA pode introduzir incertezas e piorar o desempenho em cenários onde o modelo já era altamente preciso. Portanto, embora o TTA seja uma ferramenta valiosa para aumentar a robustez do modelo em situações desafiadoras, sua aplicação deve ser cuidadosamente ponderada para evitar a degradação do desempenho em casos previamente bem-sucedidos.

6.1 References

References

Seffi Cohen, Niv Goldshlager, Bracha Shapira, and Lior Rokach. 2023. Ttanad: Test-time augmentation for network anomaly detection. *Entropy*, 25(5).

Masanari Kimura. 2024. Understanding test-time augmentation.

Divya Shanmugam, Davis W. Blalock, Guha Balakrishnan, and John V. Guttag. 2020. When and why test-time augmentation works. *CoRR*, abs/2011.11156.