


EfficientNets aplicadas à esteganálise em imagens digitais

EfficientNets applied to digital images steganalysis

Rafael Albuquerque¹

 orcid.org/0000-0001-7927-4036

Arlington Rodrigues²

 orcid.org/0000-0001-5189-820X

Gildo Ferrucio³

 orcid.org/0000-0001-9228-8870

Julia Aguiar⁴

 orcid.org/0000-0003-2756-2276

José Amarildo Filho⁵

 orcid.org/0000-0001-9203-0554

Francisco Madeiro⁶

 orcid.org/0000-0002-6123-0390

¹Fundação para Inovações Tecnológicas - FITec, Recife, Brasil. E-mail: rba@ecomp.poli.br

²Fundação para Inovações Tecnológicas - FITec, Recife, Brasil. E-mail: abr@ecomp.poli.br

³Fundação para Inovações Tecnológicas - FITec, Recife, Brasil. E-mail: gfsmd@ecomp.poli.br

⁴Fundação para Inovações Tecnológicas - FITec, Recife, Brasil. E-mail: jkbca@ecomp.poli.br

⁵Fundação para Inovações Tecnológicas - FITec, Recife, Brasil. E-mail: jalf@ecomp.poli.br

⁶Universidade de Pernambuco - UPE, Recife, Brasil. E-mail: madeiro@poli.br

DOI: 10.25286/repa.v7i2.2215

Esta obra apresenta Licença Creative Commons Atribuição-Não Comercial 4.0 Internacional.

Como citar este artigo pela NBR 6023/2018: Rafael Albuquerque; Arlington Rodrigues; Gildo Ferrucio; Julia Aguiar; José Amarildo Filho; Francisco Madeiro EfficientNets aplicadas à esteganálise em imagens digitais Revista de Engenharia e Pesquisa Aplicada, v.7, n. 2, p. 32-41, 2022.

RESUMO

Diversas arquiteturas CNN com propósito específico para esteganálise foram desenvolvidas e atingiram o estado-da-arte superando os modelos anteriores que se baseavam nas etapas de extração de características e classificação. Novos conjuntos de dados de imagens foram propostos diferenciando-se dos anteriores pela quantidade de instâncias e a variação de características importantes como fator de qualidade e a carga útil (*payload*) de mensagem escondida em imagens. Além disso, novas arquiteturas de propósito geral têm se mostrado aplicáveis no âmbito da esteganálise e se beneficiam de *transfer learning* para acelerar o treinamento. Este trabalho aborda o treinamento da *Steganalysis Residual Network* (SRNET) com inicialização aleatória dos pesos e realiza a comparação de desempenho entre as arquiteturas de CNN *Efficientnet* e *Efficientnetv2*, com este último sendo 32% mais rápido que a *EfficientnetB4*, para cada época de treinamento. Por fim, também é apresentado um experimento envolvendo treinamentos sucessivos entre a imagem de cobertura e suas respectivas estego-imagens.

PALAVRAS-CHAVE: Esteganálise; deep learning; CNN;

ABSTRACT

Several CNN architectures with a specific purpose for steganalysis were developed and reached the state-of-the-art, surpassing the previous models that were based on the feature extraction and classification steps. New image datasets were proposed, differing from the previous ones by the number of instances and the variation of important characteristics such as the quality factor and the payload of hidden messages between images. In addition, new general-purpose architectures are applicable in the scope of steganalysis and benefit from transfer learning to accelerate training. This work presents the training of the Seteganalys Residual Network (SRNET) with random initialization of weights and performs the performance comparison between the CNN architectures Efficientnet and Efficientnetv2, with the latter performing 32% faster than EfficientnetB4, for each training epoch. Finally, an experiment involving successive training within the cover image and its respective stego-images.

KEY-WORDS: Steganalysis; deep learning; CNN;

1 INTRODUÇÃO

Com o avanço do uso de tecnologia nas comunicações, tem se tornado usual o compartilhamento de arquivos digitais, sejam imagens, áudio, vídeos ou texto.

Técnicas de criptografia [1] são utilizadas para proteção do conteúdo existente nas mensagens, de modo que, ainda que a mensagem seja interceptada, ela não seja inteligível, por estar cifrada.

Uma outra possibilidade consiste em dificultar a detecção da comunicação através da ocultação de informação. Esta técnica é conhecida como *information hiding* ou *data hiding* [2] (ocultação de informações ou ocultação de dados, em uma tradução livre). Esteganografia é a arte de esconder informação de maneira que não se perceba nem mesmo a existência da mensagem escondida. O processo de detecção da esteganografia é denominado esteganálise [3]. Tal processo de detecção, no âmbito deste trabalho, limita-se a realizar a classificação binária das imagens em imagem de cobertura (imagem original, sem informação escondida) e estego-imagem (imagem modificada, portadora de mensagem escondida) - sem revelar o conteúdo das informações para os casos em que for detectada.

Em arquivos de imagens digitais, a esteganografia pode ser feita nos domínios do espaço ou da frequência [4]. O domínio do espaço trata-se de alterações nos *pixels* propriamente ditos da imagem digital. O domínio da frequência diz respeito a modificações realizadas na transformada da imagem, como por exemplo, a transformada discreta dos cossenos (*Discrete Cosine Transform - DCT*).

A partir de 2014 técnicas de *deep learning* começaram a ser utilizadas para a realização de esteganálise através de classificação binária de imagens em imagem de cobertura e estego-imagem [5]. Em 2015 foi proposto o primeiro trabalho [6] de que se tem notícia envolvendo o uso de *Convolutional Neural Networks* (CNN) [7] para esteganálise de imagens.

De acordo com [8], em uma análise dos artigos publicados sobre esteganálise utilizando CNN, uma quantidade considerável de trabalhos se utilizava de bases como *Break Our Steganographic System* (BOSS) [9], *Break Our Watermarking System* (BOWS) [10] e *Break Our Watermarking System 2* (BOWS-2) [11] para realizar os experimentos. Estas bases são constituídas por imagens geradas em

condições muito controladas como, por exemplo, fator de qualidade JPEG constante e taxa fixa de ocultação de mensagem na estego-imagem.

Com o objetivo de prover um conjunto de imagens com maior diversidade de características e, com isso, possibilitar pesquisas envolvendo esteganálise com imagens semelhantes às que são utilizadas no cotidiano (onde as imagens compartilhadas possuem diversas características diferentes entre si), foi proposta a base de dados Alaska [8] e, posteriormente, uma versão ampliada, a Alaska #2 [12].

Este trabalho visa avaliar a utilização da SRNET [13] - reconhecida como o estado-da-arte para classificação binária de imagens com esteganografia na frequência - e realizar uma comparação de desempenho entre a *EfficientNet* [14] e a *EfficientNetv2* [15], arquiteturas de CNN de propósito geral, propostas pelo *Google*.

Este trabalho também apresenta algumas dificuldades e soluções encontradas para realização de simulações com uso da SRNET no âmbito de esteganálise.

As próximas seções deste documento estão organizadas da seguinte forma: a **Seção 2** apresenta uma visão geral sobre a arquitetura das principais CNNs desenvolvidas com propósito específico para esteganálise. A **Seção 3** apresenta as arquiteturas de CNN utilizadas para os experimentos deste trabalho. A **Seção 4** discorre sobre as principais dificuldades encontradas, como estas impactam diretamente o treinamento dos modelos e a seleção de alguns parâmetros e finaliza apresentando algumas das soluções escolhidas. A **Seção 5** traz os experimentos e seus resultados, além de uma discussão sobre os resultados encontrados e possibilidades para experimentações futuras. A **Seção 6** finaliza o artigo apresentando algumas das conclusões e considerações finais sobre o trabalho.

2 CNN EM ESTEGANÁLISE

O uso de esteganálise para classificação binária de imagens é relativamente recente e a maioria dos trabalhos foi publicado nas últimas duas décadas. Vários paradigmas e técnicas [16] têm sido propostos.

Especialmente a partir de 2016, tem se intensificado o uso de técnicas de *deep learning* [17], principalmente com uso de CNN, por permitir que a extração de características das imagens seja realizada sem a estrita necessidade de

conhecimento profundo em esteganálise. Uma característica importante das CNNs é o ajuste dos pesos, baseado na taxa de erro, através do mecanismo de retropropagação.

Dentre as principais arquiteturas propostas recentemente e que se tornaram referência na área destacam-se:

- QIAN-Net [6], proposta em 2015;
- XU-Net [18], proposta em 2016;
- YE-Net [19], proposta em 2017;
- YEDROUDJ-Net [20], proposta em 2018;
- SRNET [13], proposta em 2018;
- ZHU-Net [21], proposta em 2019.

Considerando as datas de proposição de cada modelo e analisando suas alterações é possível perceber que, a cada nova arquitetura proposta, elementos de sucesso da anterior eram incorporados. Um exemplo disto é a utilização de *Batch Normalization* [22], mudanças na função de ativação ou até mesmo a utilização de operações convolucionais mais modernas como *Depthwise Convolutions* e *Depthwise Separable Convolutions*.

3 ARQUITETURAS CNN UTILIZADAS

Para os experimentos realizados neste trabalho foi utilizada uma variedade de arquiteturas. Ao término de uma série de experimentos exploratórios, e com base nas informações presentes nas publicações da área, foram escolhidas a SRNET, *EfficientNet* e *EfficientNetv2*.

3.1 SRNET

A SRNET foi a CNN utilizada na maioria dos testes iniciais por ser considerada, na literatura, como o estado-da-arte para esteganálise com imagens na frequência – que é o escopo deste trabalho – e, dado que o seu *design* não utiliza nenhuma estratégia de inicialização de pesos através de técnicas de extração de características – como filtros passa-alta ou SRM, foi a que consumiu mais tempo entre experimentações e análises. Algumas destas dificuldades e soluções estão descritas na Seção 4.

3.2 EfficientNet

A *EfficientNet* [14] é uma CNN de propósito geral para operações envolvendo classificação de imagens. Uma outra contribuição do Google, ao propor essa rede, foi definir uma estratégia de permitir escalar as redes com base em alguns parâmetros como: *largura* (que corresponde à

quantidade de filtros aplicados em uma camada de convolução), *profundidade* (referente ao número de camadas da rede) e *resolução* (densidade de *pixels* da imagem de entrada).

Com base nesta estratégia de escalonamento das redes foram criadas oito versões desta arquitetura intituladas *EfficientNetB0* a *EfficientNetB7*, conforme a Tabela 1, que apresenta os modelos e suas respectivas quantidades de parâmetros.

Tabela 1 – Arquiteturas de CNN da família *Efficientnet* e a quantidade de parâmetros de cada uma delas.

Arquitetura	Qtde. parâmetros
EfficientNetB0	5,3M
EfficientNetB1	7,9M
EfficientNetB2	9,2M
EfficientNetB3	12,3M
EfficientNetB4	19,5M
EfficientNetB5	30,6M
EfficientNetB6	43,3M
EfficientNetB7	66,7M

Fonte: Os autores

3.3 EfficientNetv2

A *EfficientNetv2* [15] é uma evolução da *EfficientNet* desenvolvida pelos mesmos autores com o intuito de obter uma maior velocidade de treinamento e melhoria na eficiência dos parâmetros.

A principal mudança de arquitetura se deve ao uso de um novo bloco chamado *Fused-MBConv* [23] nas primeiras camadas que substitui a convolução de dimensão 1x1 para expansão, seguida da *DepthwiseConv*, por única convolução de expansão 3x3. Além disso a *EfficientNetv2* não faz uso de filtros 5x5 como a *EfficientNet*, mantendo sempre a dimensão 3x3. Outra mudança importante é que o escalonamento das dimensões da *EfficientNetv2* é feito de maneira não uniforme.

4 PROBLEMAS E DIFICULDADES

Durante as simulações foi verificada uma série de dificuldades que, em sua maioria, tinha como causa a escassez de recursos computacionais e a utilização de plataformas em *cloud* como o *Kaggle* e *Colab*. Para facilitar a exposição, em todas as seções deste artigo, onde houver referências às plataformas em *cloud*, consideram-se o *Kaggle* e o *Colab*.

4.1 Recursos de Hardware

A limitação de recursos de hardware impacta diretamente no ajuste de alguns parâmetros como o *batch size*, por exemplo, e pode inviabilizar a execução dos testes por deixar o tempo de cada época consideravelmente longo, mesmo com a utilização de GPUs ou TPUs.

4.1.1 Memória

Em bases menores, como a CIFAR-10 [24], que apesar de ter 50 mil imagens de treinamento e 10 mil imagens de testes, é formada por imagens em baixa resolução (32x32), é possível carregá-la inteiramente em variáveis e efetuar o treinamento com diversas configurações de *batch size* porque os recursos disponíveis nas plataformas em *cloud* utilizadas são suficientes.

As bases mais utilizadas para trabalhos envolvendo esteganálise, a saber, BOSS, BOWS e BOWS-2, contemplam 10 mil instâncias, 4 mil instâncias e 10 mil instâncias, respectivamente. Além disso, apresentam imagens com maior resolução que a CIFAR-10, sendo utilizadas, em alguns trabalhos, com resolução 256x256 e em outros 512x512, mas em escala de cinza. Com essas características, ainda é possível, a depender da quantidade de memória disponível, carregar a base inteiramente em variáveis e efetuar o treinamento de forma análoga à descrita anteriormente para a CIFAR-10.

Em se tratando da Alaska #2, temos um acréscimo significativo tanto na resolução das imagens, tendo em vista que elas são 512x512x3, quanto por serem coloridas. Além disso, esta base possui uma quantidade bem maior de amostras, totalizando 300 mil. Estas características da base exigem uma capacidade de memória maior, por também exigirem uma arquitetura de rede mais densa e, por conseguinte, imprimem um nível de dificuldade elevado para realizar os treinamentos nos ambientes em *cloud* utilizados devido às restrições de tempo para uso impostas por essas plataformas.

4.1.2 Processamento

A capacidade de processamento disponível pode se tornar essencial para a viabilidade do experimento. A quantidade de instâncias no conjunto de treinamento, a resolução das instâncias e a quantidade de parâmetros treináveis na arquitetura de rede CNN utilizada para classificação

são alguns dos parâmetros que influencia o custo computacional exigido.

Esforços têm sido empreendidos para o desenvolvimento de novas operações convolucionais que sejam mais eficientes e utilizem menos recursos computacionais, como por exemplo a *Depthwise Convolution*. Além disto, novas arquiteturas de rede têm sido propostas buscando um equilíbrio entre o número de parâmetros, filtros, camadas e profundidade e o impacto disto nos resultados obtidos.

4.2 Batch Size

O parâmetro *batch size* define a quantidade de instâncias – neste trabalho, as imagens - que serão apresentadas à rede CNN a cada iteração, também chamada *step*. Após o processamento do número de instâncias definido neste parâmetro, os pesos da rede serão atualizados e um novo *batch size* é processado. Dessa forma, quanto maior o valor deste parâmetro maior a necessidade de memória para alocar as instâncias e mais rápido ocorrerá o treinamento, devido a menor quantidade de *steps* necessários para iterar em todo o lote de imagens. Em contraponto, um *batch size* de valor pequeno utilizará uma maior quantidade de *steps*, resultando em um tempo maior de treinamento para cada época.

Nos testes iniciais, por limitações de memória, utilizou-se um *batch size* de valor 4 para treinamento da SRNET. Isto levou a duração de uma época a pouco mais de sete horas de duração, considerando um conjunto de treinamento de 264 mil instâncias.

Esse parâmetro impacta não apenas o tempo necessário para realizar cada época de treinamento, mas também a quantidade de épocas necessárias.

4.3 Alternativas e soluções

Como visto na Seção 4.1, limitações de *hardware* foram relatadas. Esta seção apresenta soluções utilizadas durante os experimentos.

4.3.1 Gradient Accumulation

As limitações de memória tornaram necessária a utilização de um *batch size* de valor pequeno, entre 4 e 8. Em se tratando da SRNET seus autores declaram que deve ser utilizado valor igual ou superior a 32.

Uma solução para este problema foi a utilização de *gradient accumulation*. Funcionando como um *wrapper* para o otimizador selecionado para o treinamento, o *gradient accumulation* adiciona um novo parâmetro, o *accum_steps*. Este parâmetro define a quantidade de *steps* que o treinamento deve realizar antes que os pesos sejam atualizados. Com isso, é possível contornar a limitação de memória, que não permitia a utilização de um *batch size* de valor mais alto. Realizar um treinamento com *batch size* 8 e *accum_steps* 8, por exemplo, significa atualizar os pesos da rede a cada 64 instâncias apresentadas.

4.3.2 Uso de TPU com *tfRecords*

Uma forma de apresentar os dados de treinamento à rede é utilizando um tipo de dado do *Tensorflow* chamado *tf.Dataset*.

A solução é mais eficiente do que o uso de geradores de dados, porém pode ter desempenho prejudicado por um gargalo provocado pela leitura de dados realizado pela TPU. Operações de leitura/escrita de dados são fortemente impactadas pelas etapas de abertura do ponteiro para os arquivos. No caso da Alaska #2, ao realizar um treinamento com um conjunto de dados de 264 mil imagens, está sendo gerado este mesmo número de operações de abertura de ponteiro para leitura do arquivo – o que acarreta um gargalo e não permite pleno uso das capacidades da TPU.

Neste cenário, o tempo de treinamento de cada época foi reduzido significativamente para 2h30min considerando que o mesmo treinamento realizado na GPU consumia mais de 7h. Toda a utilização de TPUs neste trabalho se deu no *Kaggle*, uma vez que este ambiente disponibiliza TPU3v8, que contempla uma TPU de 8 núcleos com 16 GB cada.

Embora a redução tenha sido significativa, ainda havia margem para redução. O gargalo criado pelas operações de leitura/escrita dos arquivos pode ser contornado pela utilização de *tf.Records*. Este tipo de dados do *Tensorflow* possibilita a criação de registros de dados contendo várias imagens, bem como a associação de outras informações relevantes a cada uma delas como, por exemplo, os rótulos. Para a realização dos experimentos deste artigo foram criadas as bases de dados descritas na Seção 5.2.

5 EXPERIMENTOS

Os experimentos foram realizados utilizando as redes SRNET, *EfficientNet* e *EfficientNetv2*.

5.1 Métricas

Vale ressaltar que a base ALASKA #2 não é balanceada para classificação binária, pois para cada imagem de cobertura há 3 exemplos de estego-imagens. Uma forma de tratar este desbalanceamento é fazer classificação multiclasse. No caso da ALASKA2 esta classificação deve apontar se a imagem é: imagem de cobertura, estego-imagem do tipo JMIPOD, estego-imagem do tipo UERD ou estego-imagem do tipo J-UNIWARD.

Contudo, a competição que promove a base ALASKA2 adota como métrica a wAUC (*weighted Area Under Curve*, ou Área Ponderada Embaixo da Curva, em tradução livre). Esta métrica, conforme definida em [12], permite medir a acurácia geral, levando em consideração que quanto menos falsos-positivos o modelo apontar, melhor seu desempenho. Logo, boa parte dos trabalhos segue esta métrica como base de comparação [25][26][27][28].

5.2 Dataset

Para realização dos testes descritos nesta seção, foram criadas duas bases de dados^{1,2} utilizando *tfDataset* com registros do tipo *tfRecords*. Cada registro foi criado contendo 1000 instâncias do conjunto. Informações específicas e detalhes sobre os descritores e como utilizar a base podem ser encontradas no ambiente do *Kaggle* visto que as duas bases foram disponibilizadas publicamente para uso irrestrito.

As duas bases de dados geradas diferem na forma em que foram divididas para formação de seus respectivos conjuntos de treino, validação e teste e, além disto, no tipo de treinamento que a possibilitam fazer.

A base de dados intitulada *alaska2-tfrecord512x512train-val-test* foi gerada com o objetivo de realizar o treinamento contemplando a imagem de cobertura e as três versões de estego-imagem disponíveis no mesmo *batch* (lote). Das 300 mil imagens disponíveis, considerando as 75

¹ <https://www.kaggle.com/rbaslack/alaska2-tfrecord512x512train-val-test>

² <https://www.kaggle.com/rbaslack/alaska2tfrecord-512x512pairconstraint>

mil imagens de cada classe, foram selecionadas 264 mil imagens para treino, 12 mil imagens para validação e 24 mil imagens para teste.

A base de dados intitulada *alaska2tfrecord-512x512pairconstraint* possui três conjuntos de treinamento, três conjuntos de validação e um conjunto de teste. Cada conjunto de treinamento e validação contempla uma imagem de cobertura e sua respectiva estego-imagem, da seguinte forma:

- treino 1 e validação 1 contemplam as imagens de cobertura e a respectiva estego-imagem utilizando o algoritmo JMIPOD.
- treino 2 e validação 2 contemplam as imagens de cobertura e a respectiva estego-imagem fazendo uso do algoritmo J-UNIWARD.
- treino 3 e validação 3 contemplam as imagens de cobertura e a respectiva estego-imagem com a mensagem oculta gravada com o algoritmo UERD.

Das 150 mil imagens existentes para cada conjunto, foram utilizadas 128 mil imagens para treinamento e 10 mil para validação. O conjunto de testes é formado por 24 mil imagens, sendo 6 mil imagens de cobertura e 18 mil estego-imagem.

5.3 SRNET

A SRNET é considerada o estado-da-arte para realização de esteganálise de imagens com o uso de esteganografia na frequência. Devido a isto e à sua característica de não utilizar nenhuma estratégia de ajuste inicial dos pesos, como as demais arquiteturas citadas na seção 2 deste trabalho adotam, esta rede se tornou a primeira escolha para implementação e análise neste trabalho.

Entretanto, em sua versão padrão, não foi obtido sucesso em seu uso. A rede não apresentou melhoria nos resultados obtidos ao longo das épocas de treinamento.

Inicialmente surgiram dificuldades no escalonamento do hiperparâmetro *batch size*, conforme citado anteriormente. O indicado pelos autores desta arquitetura é a configuração deste parâmetro com valores superiores a 32. Isto não era possível por limitação de memória nas plataformas utilizadas - com o limite de configuração possível variando entre 4 e 8. A solução foi a utilização de *Gradient Accumulation* (Seção 4.3.2) e, além disto, a utilização de TPU em detrimento da GPU.

Em se tratando dos otimizadores foram testados o *Adam* [29], o *Adamax* [29], utilizado no artigo da SRNET e declarado por seus autores como o mais estável e rápido para convergência durante os

experimentos realizados por eles, além do *AdamW* [30] – frequentemente utilizado nos artigos mais recentes envolvendo esteganálise e a base Alaska #2. Importante destacar que os testes realizados quando a SRNET foi proposta utilizavam bases como BOSS, BOWS e BOWS-2. Nos experimentos mais recentes descritos pelos autores da SRNET [26][25] também tem sido utilizado o *AdamW*.

Em relação à taxa de aprendizado, foram realizados testes utilizando valores fixos (LR=0,001) e diversos *callbacks* para controle dinâmico deste parâmetro. Dentre os principais pode-se destacar o uso do *LearningRateScheduler* e *ReduceLROnPlateau*. O primeiro na tentativa de definir, através de observação, taxas de aprendizado diferenciadas contemplando uma fase de “crescimento acelerado”, uma “fase de estabilização” e uma fase de “ajuste fino”, enquanto o segundo foi configurado com um valor inicial LR=0,001, um fator de redução de 0,9, o atributo *patience* definido como 7 épocas e um valor mínimo aceitável para o LR definido como 1e-6.

Ao treinar a rede com classificação binária, realizando o agrupamento das 3 classes estego-imagens em apenas 1 classe, estego, o resultado após a execução do treinamento em várias épocas era 0,75 de acurácia e 0,5 AUC.

5.4 SRNET com SRM

Baseando-se na experiência de redes como a YEDROUJ-NET e YE-NET foi realizada uma alteração na SRNET adicionando uma camada de pré-processamento ao modelo. Esta camada utiliza uma operação de convolução 5x5 com os 30 filtros SRM para inicialização de pesos e fazendo uso da função de ativação tangente hiperbólica.

Com a adição desta camada, a SRNET começa a convergir nos testes realizados, a partir da terceira época. Utilizando um modelo de classificação multiclasse com 4 classes e função de ativação *softmax*, percebe-se uma evolução considerável nas primeiras 20 épocas, atingindo valores de acurácia entre 0,52 e 0,55 enquanto o AUC atinge valores próximos a 0,62.

Não são valores tão significativos quanto os que foram obtidos com outras redes, mas esse resultado reforçou a hipótese de que o problema de convergência pode ter origem na inicialização aleatória dos pesos especificamente para esteganálise nesta base devido às suas características.

Em se tratando do problema da convergência e inicialização dos pesos, pretende-se continuar as experimentações do uso desta rede com uma base mais simples como a BOSS e utilizar suas imagens de cobertura originais para gerar estego-imagens fixando parâmetros como fator de qualidade JPEG e taxa bpnzac e apenas um dos algoritmos de esteganografia na frequência. Com isto, espera-se realizar *transfer learning* (transferência de aprendizado, em uma tradução livre) e efetuar o treinamento a posteriori na base da Alaska #2, beneficiando-se do prévio ajuste de pesos realizado neste experimento com a BOSS.

Além disto, propõe-se alterar o processo de decodificação das imagens JPEG presente nos *tfRecords*. Atualmente utiliza-se a função do *tensorflow decode_jpeg()* e esta função efetua o arredondamento dos valores dos *pixels* gerados a partir da multiplicação das tabelas DCT e matrizes de quantização. Considerando que o processo de ocultação da mensagem na imagem objetiva realizar o mínimo possível de modificações na imagem para que as mudanças não sejam perceptíveis, gravar os pixels com valores inteiros, arredondando os dados, pode ter repercussão no processo de classificação.

No trabalho da SRNET os autores declaram a utilização de valores de *pixel* sem arredondamento, mas sem fazer menção explícita de que há algum benefício trazido por esta ação.

5.5 Efficientnet e Efficientnetv2

Diante das dificuldades enfrentadas com o treinamento e convergência da SRNET optou-se pela utilização de arquiteturas de redes que disponibilizassem um ajuste de pesos inicial para que o treinamento e classificação para esteganálise pudesse se beneficiar do aprendizado anterior através de *transfer learning* [31][32].

Dentre os mais recentes estudos algumas arquiteturas de CNN de propósito geral têm sido testadas [26][25], sobretudo as treinadas utilizando a base de classificação *ImageNet* [33] e que possuem o ajuste de pesos utilizado neste treinamento disponíveis para uso.

Nesta seção descrevem-se os resultados de alguns dos experimentos realizados utilizando tanto a *EfficientNet* quando a *EfficientNetv2*.

5.5.1 Multiclasse

O problema de classificação binária foi transformado em multiclasse contemplando quatro

classes: a imagem de cobertura e os três tipos de estego-imagem de acordo com o algoritmo utilizado.

Os parâmetros utilizados foram:

- *Batch size* 512 (64 * 8 núcleos de TPU);
- AdamW, LR 1e-3 e *weight decay* 1e-5;
- Inicialização de pesos *imagenet*;
- *ReduceLRonPlateau* configurado com atributo *patience* 7, fator de multiplicação definido em 0,6 e um LR mínimo definido como 1e-6.
- Função de ativação *softmax* na camada de classificação.

Os resultados estão apresentados na Tabela 2, onde podem ser verificados: a Arquitetura CNN utilizada (coluna Arq. CNN), o tempo médio necessário para o modelo realizar o treinamento de uma época (coluna Tempo médio por época), o número da época em que se obteve o melhor resultado de acordo com as métricas selecionadas (coluna Melhor época) e os valores das métricas selecionadas (coluna Acurácia e coluna AUC).

Tabela 2 – Resultado dos experimentos utilizando as arquiteturas de rede *EfficientNet* e *Efficientnetv2*, considerando o máximo de 60 épocas.

Arq. CNN	Tempo médio por época	Melhor época	Acurácia	AUC
B0	450s	55	0,6888	0,9212
B2	590s	50	0,6881	0,9225
B3	716s	59	0,7071	0,9308
B4	912s	58	0,7062	0,9296
B7	2100s	30	0,7127	0,9357
V2-S	620s	56	0,7054	0,9289

Fonte: Os autores

Os trabalhos realizados com a Alaska #2 e a *EfficientNet* utilizam apenas a métrica AUC como resultado. Ao comparar este indicador, é possível perceber que os resultados obtidos estão ligeiramente aquém dos publicados (-0,02 AUC). As principais hipóteses para esta diferença advêm do arredondamento dos valores dos *pixels* durante o processo de decodificação do JPEG e também por estarmos testando as versões padrão da *EfficientNet*, sem realizar as alterações de retirada das operações de *pooling* das primeiras 7 camadas, redução do *stride* dos *average pooling* para 1 e a alteração das funções de ativação, conforme realizado em [26].

Apesar disto, a inclusão da *Efficientnetv2-S* amplia as possibilidades ao compará-la com as arquiteturas da *Efficientnet*. Não é de conhecimento dos autores deste trabalho a existência de estudo

semelhante envolvendo esteganálise e a utilização da arquitetura *EfficientNetV2*.

Comparando os valores de acurácia e AUC e do tempo necessário para treinamento de cada época da *EfficientnetB4* e da *Efficientnetv2-S*, é possível comprovar nas condições de treinamento detalhadas neste artigo que os modelos da versão 2 apresentam resultado semelhante à versão anterior desta rede considerando as métricas utilizadas. Porém, possuem uma maior eficiência na realização do treinamento, conforme declarado por Tan *et al* em [15]. O modelo *Efficientnetv2-S* possui aproximadamente 3 milhões de parâmetros a mais que o *Efficientnet-B4* e foi 32% mais rápido (tempo médio por época).

5.5.2 Classificação binária

Um outro experimento proposto é a realização de treinamentos sequenciais utilizando a base de dados *alaska2tfrecord-512x512pairconstraint*, descrita na Seção 5.2. Os parâmetros utilizados são os mesmos descritos para o experimento anterior, Seção 5.5.1.

Foi utilizada a *EfficientnetV2-S* para este experimento. Ao realizar a primeira etapa do treinamento, foram obtidos os resultados apresentados na Tabela 3, onde são apresentados: o subconjunto de treino e validação utilizados neste treinamento (coluna Conj.), as respectivas classes pertencentes a cada subconjunto (coluna Classes), o número da época que apresentou melhor resultado de acordo com as métricas selecionadas (coluna Melhor época) e os valores das métricas selecionadas utilizando os respectivos conjuntos de validação (coluna Acurácia e coluna AUC).

Tabela 3 – Resultado dos experimentos da etapa 1 utilizando a arquitetura *EfficientNetv2-S* para classificação binária entre imagem de cobertura e estego-imagem, considerando o máximo de 65 épocas.

Conj.	Classes	Melhor época	Acurácia	AUC
1	Cobertura e JMiPOD	51	0,8693	0,9544
2	Cobertura e J-UNIWARD	42	0,7217	0,8301
3	Cobertura e UERD	60	0,8591	0,9472

Fonte: Os autores

É possível verificar que nesta base de dados, e de acordo com os parâmetros utilizados neste treinamento, a maior dificuldade para classificação envolve as imagens utilizando o algoritmo de esteganografia J-UNIWARD.

Com o modelo obtido nesta primeira etapa, para cada conjunto, foram realizadas mais duas etapas

envolvendo as outras possibilidades de imagem de cobertura e estego-imagem. O melhor modelo de cada etapa era utilizado para continuar a etapa subsequente. Desta forma, foram realizadas três séries de treinamento da seguinte forma:

1. {Cobertura, JMiPOD}, {Cobertura, UERD} e {Cobertura, J-UNIWARD};
2. {Cobertura, J-UNIWARD}, {Cobertura, JMiPOD} e {Cobertura, UERD};
3. {Cobertura, UERD}, {Cobertura, JMiPOD} e {Cobertura, J-UNIWARD}.

A Tabela 4 destaca que o melhor resultado é obtido através do treinamento da série composta pelo treinamento sucessivo dos conjuntos 1, 3 e 2.

Para o treinamento de cada conjunto foi utilizado o melhor modelo obtido na fase de treinamento anterior.

Por fim, foi utilizado o modelo que gerou o melhor resultado de predição com o conjunto de teste (destacado na Tabela 4), para efetuar uma nova etapa de treinamento. Desta vez, contemplando a imagem de cobertura e suas respectivas três estego-imagens.

Tabela 4 – Resultado da predição utilizando o melhor modelo obtido no treinamento após as três etapas utilizando o conjunto de testes.

Série	Acurácia	AUC
1, 3, 2	0,6877	0,8054
2, 1, 3	0,6346	0,7094
3, 1, 2	0,6344	0,7160

Fonte: Os autores

A Tabela 5 apresenta os resultados do treinamento da etapa final. A coluna LR da tabela indica se o parâmetro *learning-rate* foi redefinido para LR=0,001 no início do treinamento - embora o modelo mantenha o uso de *ReduceLRonPlateau* - ou se continuou a usar o LR dinamicamente calculado desde a primeira época deste experimento em treinamentos sucessivos. A redefinição deste parâmetro no início desta quarta etapa contribuiu para que o modelo atingisse um melhor resultado, em uma menor quantidade de épocas.

Tabela 5 – Resultado dos experimentos da etapa final contemplando a imagem original e suas respectivas três estego-imagens, considerando o máximo de 40 épocas no intervalo de 196 a 230.

LR	Melhor época	Acurácia	AUC
Reinício	215	0,8217	0,9149
Sem reinício	219	0,8043	0,8968

Fonte: Os autores

O modelo com reinício de LR quando submetido à predição do conjunto de testes resulta em 0,8222 para acurácia e 0,9151 para AUC.

Para efeito de comparação, foi realizado o treinamento de mais um modelo, com os mesmos parâmetros listados no início desta seção, mas sem efetuar os treinamentos sucessivos aos pares, contemplando o treinamento da imagem de cobertura com as suas três respectivas estego-imagens no mesmo *batch*.

O melhor modelo deste experimento utilizou 73 épocas de treinamento e resultou em 0,8277 para acurácia e 0,9185 para AUC, considerando a predição realizada no conjunto de teste. Comparando os resultados entre este modelo de treinamento e o modelo de treinamentos sucessivos, verifica-se que o primeiro utiliza uma menor quantidade de épocas (73 ante a 215) e ainda apresenta melhor desempenho no conjunto de testes.

Considerando o conjunto de parâmetros utilizados nestes experimentos é possível verificar que a realização do processo de treinamento sucessivos aos pares conforme descrito nesta seção não apresenta benefício comprovado apesar de plausibilidade teórica.

6 CONCLUSÕES

A adoção de *deep learning* como ferramenta para resolução de problemas tem sido intensificada nas mais diversas áreas do conhecimento. O uso de CNN, em especial, nos problemas envolvendo classificação de imagens, tem se mostrado eficiente e superado os resultados obtidos quando comparado às soluções clássicas.

Esforços têm sido envidados no intuito de desenvolver novas arquiteturas CNN que apresentem um sequenciamento de operações mais equilibrado entre seus componentes, distanciando-se de meramente aumentar as densidades das redes através do simples aumento da quantidade de camadas.

Neste trabalho o uso de *transfer learning* se mostrou de grande valia. O conhecimento adquirido pela rede na tarefa de classificação das imagens da *ImageNet* possibilitou que fossem obtidos valores superiores aos das redes criadas específicas para esteganálise, com uma menor quantidade de épocas/tempo de treinamento.

A avidez por recursos de hardware para atividades envolvendo significativo volume de dados, como a base Alaska #2, potencialmente

impacta o acesso à pesquisa na área por pesquisadores sem disponibilidade dos recursos necessários. Os ambientes em cloud como o *Kaggle* e o *Colab*, se bem utilizados, se apresentam como uma possibilidade de transpor estas limitações. Este artigo apresenta algumas soluções que não apenas viabilizaram os experimentos realizados, mas reduziram significativamente o tempo necessário para os treinamentos. Estas soluções podem ser aplicadas em outras áreas, não se limitando à esteganografia ou esteganálise.

Nos próximos trabalhos pretende-se realizar alterações na *Efficientnetv2-S* e *Efficientnetv2-M* visando melhorar seu desempenho na aplicação de esteganálise. Algumas destas alterações são o ajuste da primeira camada de convolução a fim de que a redução da resolução pela metade não seja efetuada e a retirada das operações de *pooling* das primeiras camadas desta rede.

REFERÊNCIAS

- [1] W. Mao, *Modern Cryptography: Theory and Practice*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2003.
- [2] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding - A Survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062-1078, 1999.
- [3] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. San Francisco, CA: Cambridge University Press, 2009.
- [4] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*. San Francisco, CA: Morgan Kaufmann, 2007.
- [5] S. Tan and B. Li, "Stacked Convolutional Auto-encoders for Steganalysis of Digital Images," *Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. APSIPA*, no. 1, 2014.
- [6] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep Learning for Steganalysis via Convolutional Neural Networks," *Media Watermarking, Secur. Forensics*, vol. 9409, no. March, p. 94090J, 2015.
- [7] Y. LeCun, Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series," *Handb. Brain Theory Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.
- [8] R. Coganne, Q. Giboulot, and P. Bas, "The Alaska Steganalysis Challenge: A First Step Towards Steganalysis 'Into The Wild,'" *IH MMSec 2019 - Proc. ACM Work. Inf. Hiding Multimed. Secur.*, pp. 125-137,

- 2019.
- [9] P. Bas, T. Filler, and T. Pevn, "Break Our Steganographic System": The Ins and Outs of Organizing BOSS," in *International Workshop on Information Hiding*, pp. 59–70, 2011.
- [10] A. Piva and M. Barni, "The First BOWS Contest (Break Our Watermarking System)," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505, p. 650516, 2007.
- [11] T. Furon and P. Bas, "Broken arrows," *EURASIP J. Inf. Secur.*, vol. 2008, pp. 1–13, 2008.
- [12] R. Cogranne, Q. Giboulot, and P. Bas, "ALASKA# 2: Challenging Academic Research on Steganalysis with Realistic Images," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–5, 2020.
- [13] M. Boroumand, S. Member, M. Chen, and J. Fridrich, "Deep Residual Network for Steganalysis of Digital Images," *IEEE Transactions on Information Forensics and Security*, v. 14, n. 5, p. 1181-1193, 2018.
- [14] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 6105–6114, 2019.
- [15] M. Tan and Q. V Le, "Efficientnetv2: Smaller Models and Faster Training," *arXiv Prepr. arXiv2104.00298*, 2021.
- [16] A. Selvaraj, A. Ezhilarasan, S. L. J. Wellington, and A. R. Sam, "Digital Image Steganalysis: A Survey on Paradigm Shift from Machine Learning to Deep Learning Based Techniques," *IET Image Process.*, vol. 15, no. 2, pp. 504–522, 2021.
- [17] M. Chaumont, "Deep Learning in Steganography and Steganalysis," *Digit. Media Steganography Princ. Algorithms, Adv.*, no. October 2019, pp. 321–349, 2020.
- [18] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural Design of Convolutional Neural Networks for Steganalysis," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, 2016.
- [19] J. Ye, J. Ni, and Y. Yi, "Deep Learning Hierarchical Representations for Image Steganalysis," vol. 12, no. 11, pp. 2545–2557, 2017.
- [20] M. Yedroudj, F. Comby and M. Chaumont, "Yedroudj-net: An Efficient CNN for Spatial Steganalysis," *2018 IEEE Int. Conf. Acoust. Speech Signal Process.*, no. April, 2018.
- [21] R. Zhang, F. Zhu, J. Liu, and G. Liu, "Depth-wise Separable Convolutions and Multi-level Pooling for an Efficient Spatial CNN-based Steganalysis," *IEEE Trans. Inf. Forensics Secur.*, vol. PP, no. August, p. 1, 2019.
- [22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International conference on machine learning*, pp. 448–456, 2015.
- [23] S. Gupta and B. Akin, "Accelerator-aware Neural Network Design Using Automl," *arXiv Prepr. arXiv2003.02838*, 2020.
- [24] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)," [Online]. Available: <http://www.cs.toronto.edu/~kriz/cifar.htm>, 2014.
- [25] J. Butora, Y. Yousfi, and J. Fridrich, "How to Pretrain for Steganalysis," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pp. 143–148, 2021.
- [26] Y. Yousfi, J. Butora, J. Fridrich, and C. Fuji Tsang, "Improving EfficientNet for JPEG Steganalysis," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pp. 149–157, 2021.
- [27] K. Chubachi, "An Ensemble Model Using CNNs on Different Domains for ALASKA2 Image Steganalysis," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2020.
- [28] Y. Yousfi, J. Butora, E. Khvedchenya, and J. Fridrich, "ImageNet Pre-trained CNNs for JPEG Steganalysis," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2020.
- [29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv Prepr. arXiv1412.6980*, 2014.
- [30] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv Prepr. arXiv1711.05101*, 2017.
- [31] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.
- [32] L. Torrey and J. Shavlik, "Transfer Learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, pp. 242–264, 2010.
- [33] O. Russakovsky et al., "Imagenet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.