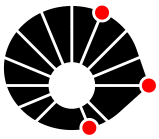

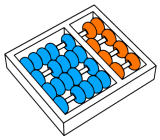

Estudo de Métodos de Aprendizado de Máquina Profundo para Interpolação de Quadros em Vídeos

H.S.L. Almeida *H. Pedrini*

Relatório Técnico - IC-PFG-22-14
Projeto Final de Graduação
2022 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Estudo de Métodos de Aprendizado Profundo para Interpolação de Quadros em Vídeos

Henrique Sandes Lima Almeida Hélio Pedrini *

Resumo

A Interpolação de Quadros em Vídeos (VFI, *Video Frame Interpolation*) é uma tarefa da Computação Visual que busca aumentar a taxa de quadros de um vídeo ao produzir novas imagens intermediárias àquelas já existentes na sequência, criando um conteúdo mais visualmente agradável aos espectadores. Embora pouco conhecido pelo público geral, este é um campo de estudo de suma importância, possuindo inúmeras aplicações práticas, tais como recuperação de vídeos baseada em conteúdo, geração de conteúdo em câmera lenta, facilitação do processo de edição de vídeos e até mesmo a intermediação de serviços de *streaming* de vídeo. O objetivo deste trabalho é estudar e avaliar o desempenho quantitativo e qualitativo de três métodos do estado da arte na área baseados em Aprendizado Profundo: RIFE (*Real-time Intermediate Flow Estimation*) [30], XVFI (*eXtreme Video Frame Interpolation*) [70] e CDFI (*Compression-Driven Flow Estimation*) [19]. Embora os modelos RIFE e XVFI tenham obtido os melhores desempenhos nas métricas utilizadas durante os testes e o modelo RIFE apresente os melhores resultados nominais em seu artigo, todas as arquiteturas demonstraram bons resultados ao gerar quadros intermediários para diferentes sequências de vídeo, com algumas faltas de detalhes muitas vezes imperceptíveis ao olho humano durante a exibição do conteúdo completo, podendo ser utilizadas como aplicações de execução eficiente em computadores de uso geral, entretanto, de alto requerimento de poder computacional de GPUs para o processo de treinamento.

1 Introdução

Interpolação de Quadros em Vídeos (VFI, do inglês *Video Frame Interpolation*) é o termo que se dá à tarefa de visão computacional de baixo nível que busca incrementar a taxa de quadros de um vídeo [3, 9, 53, 82], convertendo um conteúdo com baixa taxa de quadros (LFR, *Low Frame Rate*) para um com alta taxa de quadros (HFR, *High Frame Rate*), ao gerar um ou mais novos quadros intermediários entre dois dados quadros sucessivos da mídia, dando origem a um material de resolução temporal consideravelmente maior.

O processo da VFI dá origem a um campo de estudo em ascensão, dada sua ampla utilização em inúmeras aplicações nos domínios temporal e espacial, tais como realce da qualidade de vídeos [81], compressão de vídeos [49, 76], edição de vídeos [54], geração de conteúdo em câmera lenta [4, 34, 45, 56, 58, 63], síntese de vista [21, 22, 35, 88] e

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

recuperação de vídeos baseada em conteúdo [25, 36, 73, 74]. De uma forma geral, vídeos com uma maior taxa de quadros possibilitam uma experiência com menos instabilidade temporal e desfoque de movimento, criando um conteúdo mais visualmente agradável aos espectadores [17, 39, 40, 51]. Ademais, a utilização em tempo real de algoritmos de VFI pode permitir, por exemplo, um menor requerimento de banda de Internet para acessar serviços de *streaming* [77], o fornecimento de funcionalidades de edição de vídeo para usuários com poder computacional reduzido [54] e a adaptação da taxa de quadros do conteúdo à tela de exibição [39, 40].

Diferentemente de outros problemas no campo da visão computacional, que dependem majoritariamente de um árduo trabalho manual de anotação de dados, como a classificação de imagens, a interpolação de quadros em vídeos se beneficia muito do fato de que dados de treinamento de alta qualidade podem ser adquiridos com facilidade a partir da imensurável quantidade de vídeos disponíveis na rede, sendo necessário apenas separá-los quadro a quadro. Devido a esse fato, juntamente com sua vasta importância prática, o campo de pesquisa em VFI se encontra em franca expansão e vários algoritmos vêm sendo propostos com o intuito de aumentar a taxa de quadros em vídeos [4, 5, 11, 13–15, 24, 33, 34, 42, 44–46, 56–60, 67, 79, 85], intensificados pelo sucesso e crescente desenvolvimento de estruturas de Redes Neurais Profundas (DNN, do inglês *Deep Neural Networks*) para tarefas de visão computacional. Essas abordagens utilizando métodos de Aprendizado Profundo (*Deep Learning*) podem ser basicamente generalizadas em 3 categorias: baseadas em fluxo óptico ou de movimento (chamadas de *flow-based*, *optical-flow-based* ou ainda *motion-based*) [4, 5, 13, 24, 30, 32, 34, 38, 43–45, 56, 57, 60, 71, 78–80, 83, 85, 86]; baseadas em kernel (*kernel-based*) [4, 5, 10, 11, 42, 58, 59, 68]; e baseadas em fase (*phase-based*) [53, 55].

Porém, a interpolação de quadros em vídeos demonstra ser um problema laborioso devido a diversos fatores, como oclusão e grandes e complexas variações de movimentos e iluminação características de filmagens do mundo real. Devido à elevada dificuldade associada à estimativa e predição de tais comportamentos, a tarefa de interpolar quadros intermediários e obter resultados de alta qualidade ainda continua sendo muito desafiadora, mesmo com tantas pesquisas na área e os avanços de Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*) aplicadas à VFI.

No entanto, recentes propostas de algoritmos baseados em aprendizado profundo do estado da arte vêm apresentando resultados extraordinários [4, 5, 13, 15, 19, 24, 30, 34, 42, 43, 57, 60, 61, 67, 70, 79, 85, 87]. Nesse contexto, este trabalho tem como principal objetivo realizar um estudo comparativo de alguns dos mais recentes e bem-sucedidos métodos baseados em aprendizado profundo desenvolvidos para atacar o problema de interpolação de quadros em vídeos. Dessa forma, serão avaliados os desempenhos de três arquiteturas: XVFI-Net (*eXtreme Video Frame Interpolation Network*) [70], CDFI (*Compression-Driven Network Design for Frame Interpolation*) [19]; e RIFE (*Real-time Intermediate Flow Estimation*) [30]. Devido à limitação de poder computacional disponível ao desenvolvimento deste projeto, a comparação será realizada tendo em vista uma simples prova de conceito dos métodos supracitados, efetuando o treinamento e a avaliação das respectivas arquiteturas de redes neurais em um conjunto de dados de tamanho reduzido.

Esta Seção 1 inclui uma descrição do tema tratado neste documento, apresentando os desafios, as motivações e as aplicações associadas ao seu estudo, bem como os objetivos

do trabalho. A Seção 2 tece uma contextualização a respeito dos conceitos e trabalhos relacionados aos objetos de estudo. A Seção 3 apresenta a metodologia proposta, tratando de detalhes sobre os passos de execução do projeto, a aquisição e tratamento do conjunto de dados, as métricas de avaliação utilizadas e as estruturas dos métodos avaliados. A Seção 4 aborda os experimentos realizados durante o estudo e os recursos e parâmetros utilizados no processo. A Seção 5 trata dos resultados oriundos destes procedimentos, exibindo e discutindo-os. Por fim, a Seção 6 apresenta algumas conclusões acerca do projeto desenvolvido e, em seguida, são expostas as referências bibliográficas do material utilizado para o trabalho.

2 Conceitos e Trabalhos Relacionados

Esta seção aborda brevemente os conceitos básicos ligados ao tema proposto, assim como apresenta os principais trabalhos relacionados à Interpolação de Quadros em Vídeos e aos métodos estudados neste documento.

2.1 Inteligência Artificial e Aprendizado de Máquina

Os seres humanos há muito sonham em desenvolver autômatos pensantes, um desejo que se mostra presente desde milênios atrás, na Grécia antiga. A Inteligência Artificial (AI, do inglês *Artificial Intelligence*) permaneceu durante muito tempo apenas no ideário das pessoas, sofrendo influência de diversas áreas de estudo durante os anos de concepção do campo, como engenharia, biologia, teoria da comunicação, matemática, estatística, lógica, filosofia, linguística, entre outros. Apesar do artigo de Alan Turing em 1950 que introduziu a possibilidade de se programar um computador eletrônico que se comportasse de forma inteligente, incluindo a descrição do famoso Teste de Turing, o termo Inteligência Artificial, de fato, somente surgiu em 1956, na conferência *Dartmouth Summer Research Project on Artificial Intelligence*, cunhando o nome da área como parte do nome do evento com finalidade de estudo da mesma. Apenas a partir da segunda metade do século, tem-se conseguido construir máquinas com poder computacional suficiente para testar as hipóteses e demonstrar mecanismos de comportamento inteligente que só existiam como remotas possibilidades teóricas [7, 23].

Para resolver problemas no computador, precisamos de um algoritmo, isto é, uma sequência bem definida de instruções que devem ser seguidas em ordem para transformar um conjunto de dados de entrada em uma saída esperada. Porém, para algumas tarefas não conhecemos algoritmos bem definidos para resolvê-las e queremos então que a máquina seja capaz de automaticamente gerar um algoritmo capaz de resolver tais tarefas. Daí vem o conceito de AI, ou seja, para ser inteligente, um sistema precisa ter a capacidade de aprender e se adaptar em um ambiente em constante mudança, de forma que o desenvolvedor não precise prever soluções para todas possíveis situações. Entretanto, enquanto uma parte das tarefas conseguem ser facilmente interpretadas por modelos e regras matemáticas bem conhecidas pelos computadores, o verdadeiro desafio para a Inteligência Artificial se encontra em resolver problemas que são fáceis para as pessoas resolverem, mas difíceis de descrever tal solução como um conjunto de regras bem determinadas, tais como algumas

das aplicações práticas e tópicos de pesquisa mais buscados atualmente: automatização de trabalhos manuais, reconhecimento de fala e imagens, predizer diagnósticos na medicina, entre outros [1, 7, 23].

Porém, apesar de não existirem algoritmos diretos para resolver estes problemas, temos quantidades incontáveis de dados disponíveis e coletados ao longo da história que representam o comportamento esperado em diversas situações. Dessa forma, seria então necessário que os sistemas de AI fossem capazes de extrair e reconhecer padrões a partir desses dados, mesmo que não seja possível identificar precisamente o processo completo, o que deu origem à área de Aprendizado de Máquina (ML, do inglês *Machine Learning*), como uma parte dos estudos em Inteligência Artificial [1, 23].

O Aprendizado de Máquina se refere exatamente à capacidade de fazer as máquinas aprenderem a extrair um dado comportamento a partir dos dados disponíveis e de experiência prévia, sendo programados de forma a melhorar esse processo continuamente com o objetivo de otimizar um certo critério pré-definido. Dentro da área, os algoritmos desenvolvidos são primariamente divididos em três grandes grupos, classificados pela forma com a qual o sistema interage com os dados disponibilizados: Aprendizado Supervisionado (*Supervised Learning*), Aprendizado Não-Supervisionado (*Unsupervised Learning*) e Aprendizado por Reforço (*Reinforcement Learning*) [1].

No Aprendizado Supervisionado, são providos ao computador dados já rotulados, ou seja, para os quais se sabe exatamente, a partir dos dados de entrada, qual é a resposta esperada a ser gerada. Ao fornecer tais dados à máquina, a ideia do desenvolvedor é de que o sistema seja capaz de aprender a mapear os padrões nos dados de forma a determinar os fatores que levam um conjunto de informações de entrada a ser relacionado a uma dada informação de saída. É esperado, dessa forma, que o sistema seja de fato capaz de aprender os padrões presentes nos dados e não decorar as respostas esperadas, ou seja, que este tenha capacidade de generalização, podendo categorizar corretamente exemplos novos que diferem daqueles apresentados ao modelo durante seu treinamento [1, 6].

O Aprendizado Não-Supervisionado, por sua vez, já não possui mais uma figura de um "supervisor" que fornece ao sistema as saídas esperadas, sendo disponível à máquina somente os dados de entrada. Dessa forma, o objetivo para essa classe de algoritmos seria de encontrar regularidades e padrões nos dados de entrada que os relacionem, sem uma função pré-determinada, com o objetivo de, por exemplo, agrupá-los em grupos com características semelhantes ou determinar a distribuição dos dados dentro do conjunto [1, 6].

Já no Aprendizado por Reforço, é abordado o caso no qual o sistema não tem como saída uma resposta única, mas sim uma sequência de ações a serem tomadas. Dessa forma, a máquina precisa descobrir, em um processo de tentativa e erro, um conjunto de ações a realizar em determinadas situações, a fim de maximizar uma função de recompensa pré-determinada. Porém, encontrar uma política que descreva o comportamento geral da máquina em todos os momentos é, neste caso, mais importante do que maximizar a recompensa imediata nas ações de cada instante, visto que cada ação pode impactar na recompensa e no resultado de todos passos subsequentes. Assim, o algoritmo precisa avaliar o desempenho de cada sequência de ações tomadas e aprender dos conjuntos anteriores para gerar uma política de comportamento geral que se adapta bem à situação proposta [1, 6].

Neste projeto, o objetivo será gerar uma imagem intermediária a partir de duas imagens

de entrada, de forma que essa saída represente um estado de tempo intermediário às duas imagens fornecidas como parte de um vídeo. Como dispomos de um grande conjunto de dados com informações rotuladas sobre quais quadros devem ser gerados como intermediários a partir de determinados quadros dados como entrada, os algoritmos aqui estudados se enquadram dentro da classe de Aprendizado Supervisionado, para que a máquina consiga aprender como gerar a saída esperada a partir dos exemplos completos fornecidos. Em mais detalhes, as técnicas que serão avaliadas neste trabalho pertencem à categoria das Redes Neurais Profundas (DNN, do inglês *Deep Neural Networks*), um nicho dentro dos programas de ML, como ilustrado na Figura 1.

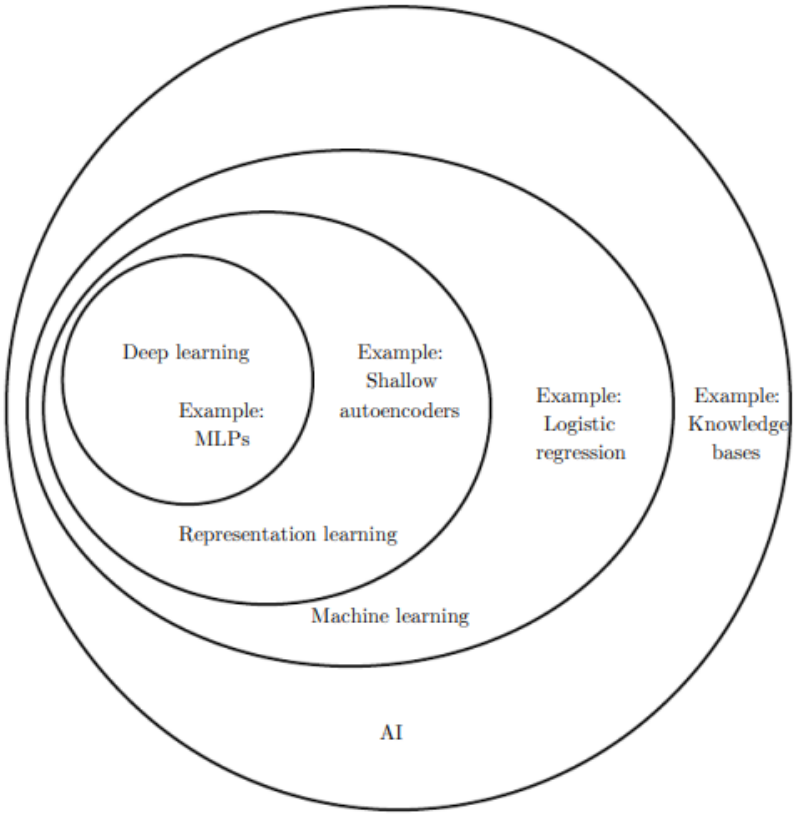

Figura 1: Diagrama de Venn mostrando as áreas de Inteligência Artificial. O Aprendizado Profundo é um tipo de algoritmo do Aprendizado de Representação, que por sua vez é um classe dentro do Aprendizado de Máquina, que compreende sistemas utilizados para alguns mas não todos problemas de Inteligência Artificial [23].

O desempenho dos algoritmos de ML depende muito da representação utilizada nos dados fornecidos ao sistema, sendo cada informação sobre uma unidade do conjunto conhecida como característica. Muitos dos problemas podem ser resolvidos ao extrair dos dados o conjunto correto de características que otimizam o algoritmo e permitem à máquina aprender com facilidade os padrões necessários, porém na maioria dos casos é difícil determinar quais

são as características que formam esse conjunto. Uma solução criada para esse problema foi utilizar os algoritmos de aprendizado de máquina não só para determinar a saída a partir dos dados de entrada, mas também aprender como representar os dados de entrada da melhor forma possível, originando o Aprendizado de Representação (*Representation Learning*), que obtém na média resultados muito superiores aos sistemas baseados apenas em representações feitas à mão. Os sistemas de Aprendizado Profundo (DL, do inglês *Deep Learning*), área pertencente ao Aprendizado de Representação e dentro da qual estão situadas as DNN, surge com o objetivo de atacar esse problema e criar estruturas com representações encadeadas, de forma que cada parte da representação dos dados de entrada é expressa em termos de outras representações, mais simples [23].

2.2 Aprendizado Profundo

Embora seja um tópico que tenha sido popularizado e esteja em plena expansão recentemente, os conceitos de Aprendizado Profundo já existem desde a década de 1940, embora o campo de estudo tenha sido conhecido por diversos nomes até ser atualmente adotado o nome Aprendizado Profundo. De forma geral, o desenvolvimento de DL pode ser dividido em três grandes ondas: DL conhecido como *cybernetics* entre as décadas de 1940 e 1960; DL conhecido como *connectionism* entre as décadas de 1980 e 1990; e o atual ressurgimento sob o nome de *Deep Learning*, iniciando em 2006 [23].

O nome da principal classe de algoritmos do Aprendizado Profundo, as Redes Neurais, surgiu devido a alguns dos mais incipientes algoritmos que reconhecemos hoje como pertencentes a esse campo, que traziam a proposta de compor modelos computacionais para simular o aprendizado biológico, isto é, modelos para estruturar como o aprendizado acontece ou pode acontecer dentro de um cérebro. Na perspectiva de que modelos de Aprendizado Profundo seriam então sistemas inspirados no comportamento do cérebro biológico, os algoritmos ficaram também conhecidos como Redes Neurais Artificiais (ANN, do inglês *Artificial Neural Networks*). Já o termo mais moderno, Aprendizado Profundo, vai além da ideia neurocientífica de simular o comportamento do pensamento humano e apela para uma ideia do aprendizado com um algoritmo composto por múltiplas camadas e níveis de representação de forma a obter uma melhor interpretação dos dados [23].

No final do século passado, já tínhamos conhecimento teórico que demonstravam que algoritmos de Redes Neurais datados da década de 1980, ou até mesmo antes, funcionavam muito bem, mas isso ainda não se mostrava aparente, devido à dificuldade de realizar experimentos práticos com tais programas, que requerem um poder computacional muito maior do que o possibilitado pelo equipamento disponível na época e um conjunto muito grande de dados, escassos e de difícil armazenamento até o momento. A situação mudou bastante no começo do século XXI, com a crescente disponibilização de uma quantidade imensurável de dados pela rede e o aumento exponencial do poder computacional, principalmente com o uso das placas de vídeo (GPUs, do inglês *Graphical Processing Units*), inicialmente pensadas para renderização de jogos, mas cuja estrutura permitia uma execução otimizada de algoritmos de Redes Neurais, tornando estes algoritmos cada vez mais relevantes. A então terceira e atual onda de pesquisa de Aprendizado Profundo, que consolidou o nome do campo de estudo, teve início em 2006 quando Geoffrey Hinton revolucionou ao mostrar

que um algoritmo de rede neural chamado por ele de *Deep Belief Network* poderia ser, de fato, treinado de forma eficiente utilizando uma estratégia denominada *greedy layer-wise pretraining* [23, 28, 84].

Dessa forma, o foco da pesquisa passou de modelos generalizados lineares de *Machine Learning* para as RNNs, possibilitando a redescoberta de diversos conceitos chaves do Aprendizado Profundo há muito deixados de lado, como o Perceptron Multicamadas (MLP, do inglês *Multilayer Perceptron*) de 1943 [52] e as Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) de 1998 [41]. Assim, o Aprendizado Profundo se estabeleceu como uma abordagem particular de ML de grande potencial, desempenho e flexibilidade, ao aprender a representar o mundo da máquina em termos de uma hierarquia encadeada de conceitos, construindo conceitos mais complexos em termos de conceitos mais simples e representações mais abstratas em termos de representações mais concretas. A ideia de profundidade do DL advém da perspectiva de construir a representação correta dos dados ao estruturar o aprendizado em diversas camadas de diferentes transformações, de forma que cada camada origina uma diferente representação para os dados sendo processados, que agregam para o conjunto de padrões conhecidos pelo sistema, originando uma espécie de Aprendizado de Representação Multinível [23, 84].

A proposta do MLP e a criação da ideia de neurônios computacionais servem como base para o desenvolvimento de algoritmos de Redes Neurais e ilustram perfeitamente o conceito do Aprendizado Profundo com os dados de entrada, na camada de entrada, as diferentes representações dadas pelas camadas escondidas, com cada nova camada adicionando uma nova perspectiva e uma maior complexidade ao modelo, e a saída do sistema, na camada de saída, como exemplificado na Figura 2. Na representação computacional da rede, cada camada guarda como parâmetros o conjunto de pesos utilizados para multiplicar a informação vinda de cada neurônio da camada anterior, assim como um *bias* que adiciona um coeficiente linear à operação. Ao executar o algoritmo, cada unidade tem sua saída calculada como a soma ponderada das saídas provenientes de todos os neurônios da camada anterior pelos seus respectivos pesos (com exceção da camada de entrada que representa os dados de entrada não processados e não possui camada anterior a si), adicionada ao *bias* e então aplica ao valor resultante uma função de ativação, tais como degrau, sigmoide, tangente hiperbólica ou RELU (*Rectified Linear Unit*), de forma a normalizar o resultado. O aprendizado de fato do modelo se dá através do processo de treinamento, no qual, a partir de um conjunto de dados rotulados e uma função de erro pré-determinada, a Rede Neural utiliza o algoritmo de *backpropagation* para calcular o erro associado aos gradientes das funções de cada neurônio e atualizar os valores dos pesos de forma a otimizar o resultado e minimizar o erro do sistema [23, 84].

Conforme são aumentados o número de camadas na arquitetura e o número de neurônios por camada, cresce a complexidade e capacidade de interpretação do modelo, entretanto, também cresce exponencialmente o número de parâmetros necessários para sua especificação e, consequentemente, o poder computacional necessário à sua execução, já que, no modelo da MLP, todas as unidades de cada camada são conectadas a todas unidades das camadas vizinhas, conhecidas como Camadas Totalmente Conectadas (*Fully-Connected Layers*). Para tipos de dados muito densos, como, por exemplo, imagens, uma simples rede neural MLP pode precisar de bilhões de parâmetros para conseguir aprender corretamente a re-

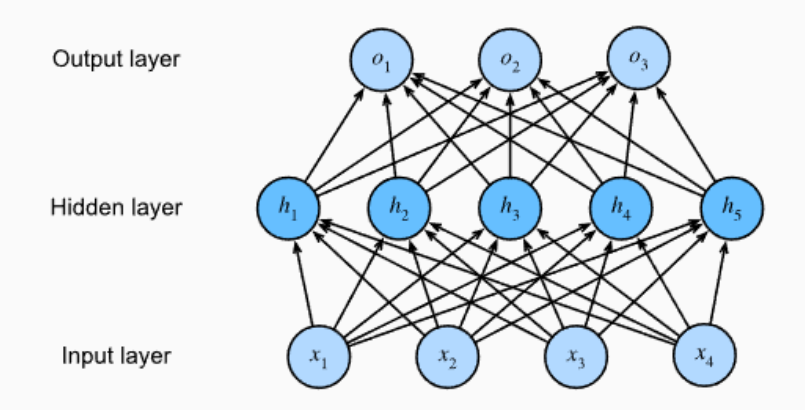

Figura 2: Exemplo de arquitetura de Rede Neural do tipo MLP (*Multilayer Perceptron*), composta por uma camada de entrada (*input layer*) com 4 neurônios, 1 camada escondida (*hidden layer*) com 5 neurônios e uma camada de saída (*output layer*) com 3 neurônios. Cada conexão entre neurônios é associada a um peso diferente, armazenado como parâmetro da estrutura [84].

presentar os padrões presentes na figura. Nesse contexto, surge a ideia de criar diferentes tipos de camadas mais especializadas, trazendo à tona diferentes operações matemáticas, para construir modelos mais simples, mais eficientes e com maior poder de generalização dos dados, como as CNNs [23, 84].

2.3 Redes Neurais Convolucionais

Ao utilizar redes MLP totalmente conectadas para processar dados de imagens, acabamos por descartar e desperdiçar a estrutura espacial da imagem no processo de transformá-las em vetores de uma dimensão para serem alimentados a cada camada, já que esse tipo de algoritmo é invariante à ordem das características do dado de entrada. Foram então desenvolvidas as Redes Neurais Convolucionais [41] precisamente com o propósito de fazer uso do conhecimento prévio de que pixels próximos uns aos outros em uma imagem normalmente possuem uma relação entre si, de forma a construir modelos que pudessem aprender a reconhecer padrões de forma mais eficiente em dados organizados em uma topologia de grade, como imagens, em 2 dimensões, ou até mesmo séries temporais, em 1D. As CNNs se provaram excepcionalmente bem-sucedidas em aplicações práticas nos últimos anos, ao ponto de que é quase impossível não considerar a abordagem para problemas relacionados a imagens, como reconhecimento, detecção de objetos ou segmentação semântica, por exemplo [23, 84].

O nome dessa classe de algoritmos advém da operação matemática utilizada em suas camadas, a convolução, um tipo de operação linear especializada no domínio do tempo ou do espaço, que busca realizar uma filtragem ao percorrer de forma sequencial os elementos da grade dos dados de entrada e centralizá-los com uma grade de pesos conhecida como filtro de convolução ou *kernel*, multiplicando o elemento centralizado e os valores em sua vizinhança pelos pesos do filtro, obtendo como saída uma nova grade, conhecida como mapa

de características (*feature map*) [62]. A fórmula para a operação de convolução entre uma imagem bidimensional I e um filtro K para um pixel na linha i e coluna j da imagem pode ser definida da seguinte forma:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{1}$$

Como a convolução é uma operação que possui a propriedade comutativa, podemos reescrever a expressão de forma equivalente como:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2}$$

Usualmente, a expressão 2 é mais direta de ser implementada por bibliotecas de Aprendizado Profundo, já que representa uma menor variação menor no conjunto de valores válidos para os índices m e n. Ademais, é importante ressaltar que a operação de convolução é normalmente confundida com a similar operação de correlação cruzada, que possui uma estrutura quase idêntica, a diferir do fato de que as somatórias da convolução são realizadas invertendo o *kernel* K em relação à entrada I. A fórmula da correlação cruzada seria então, sem a inversão:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{3}$$

A maioria das bibliotecas que disponibilizam estruturas para CNNs implementa a correlação cruzada e chamam a operação erroneamente de convolução. Porém, como parte do processo do algoritmo é aprender os pesos de cada filtro de convolução e otimizá-los para o sistema utilizado, a diferença entre as operações não possui efeito prático. A operação, ilustrada na Figura 3, prova-se muito eficiente e é comumente utilizada no processamento de imagens e sinais digitais, como, por exemplo, para a extração de determinadas características, assim como visado pelas Redes Neurais [23, 84].

Dá-se então o nome de Camadas Convolucionais às camadas de Redes Neurais que implementam a operação descrita e armazenam seus filtros associados. Utilizando-se dessas estruturas, combinadas com outras camadas e funções, as CNNs foram idealizadas e conseguiram sistematizar a ideia da invariância espacial, aplicando princípios como a invariância de translação e a localidade espacial para reconhecer padrões em qualquer posicionamento dentro de uma grade e identificar relações entre elementos próximos, de forma a aprender representações eficientes com menos parâmetros do que uma MLP, por exemplo. Como as camadas convolucionais requerem o armazenamento de um menor número de parâmetros e convoluções são operações fáceis de serem paralelizadas e rapidamente executadas em GPUs, as Redes Neurais Convolucionais conseguem ser, além de mais precisas, mais computacionalmente eficientes [84].

2.4 Estimação de Fluxo Óptico

O fluxo óptico pode ser definido como a distribuição das velocidades aparentes de movimento dos padrões de brilho em uma imagem e surge a partir do movimento relativo entre

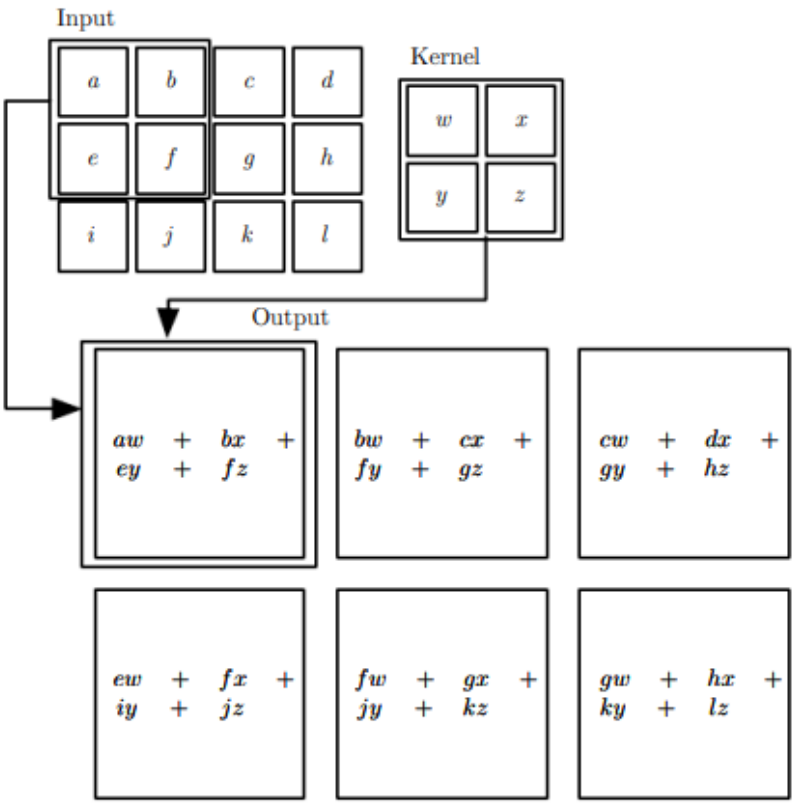

Figura 3: Exemplo uma operação de convolução sem inversão de *kernel*, ou seja, uma operação de correlação cruzada. O filtro de tamanho 2 (2 linhas e 2 colunas) "desliza" sobre as posições da grade de entrada que possui 3 linhas e 4 colunas, retornando como resultado um mapa de características com os valores apresentados na saída [23].

objetos de uma cena e um observador, o que também pode ser interpretado, dentro do contexto do estudo de Interpolação de Quadros em Vídeos, como o campo vetorial que representa os deslocamentos ocorridos entre dois quadros consecutivos de uma sequência. Dessa forma, o fluxo óptico desponta como um fator essencial à tarefa de VFI, já que pode fornecer informações importantes sobre o arranjo espacial dos objetos observados na cena e a frequência de mudança dessa arrumação, indicando regiões potencialmente pertencentes a diferentes objetos a partir de descontinuidades no campo [29].

Porém, não é possível computar o fluxo óptico em um dado ponto da imagem de forma independente dos pontos vizinhos sem a adição de limitantes adicionais ao cálculo, visto que o campo de velocidades em cada pixel possui duas componentes, enquanto a mudança no brilho de um dado ponto no plano da imagem devido a movimento somente se relaciona a uma [29]. Assim, inspirado pelo sucesso dos algoritmos de Aprendizado Profundo em tarefas de visão computacional de alto nível, como classificação de imagens, Dosovitskiy et al. [20] propuseram dois modelos de Redes Neurais Convolucionais para estimar o fluxo

óptico, FlowNetSimple e FlowNetCorr, cujas arquiteturas são ilustradas na Figura 4, introduzindo uma completa mudança de paradigma na área. O trabalho realizado demonstrou a viabilidade de estimar o fluxo óptico diretamente de imagens sem processamento prévio, enquadrando-o como um problema de aprendizado ao fazer uso do potencial de simples arquiteturas CNN [72].

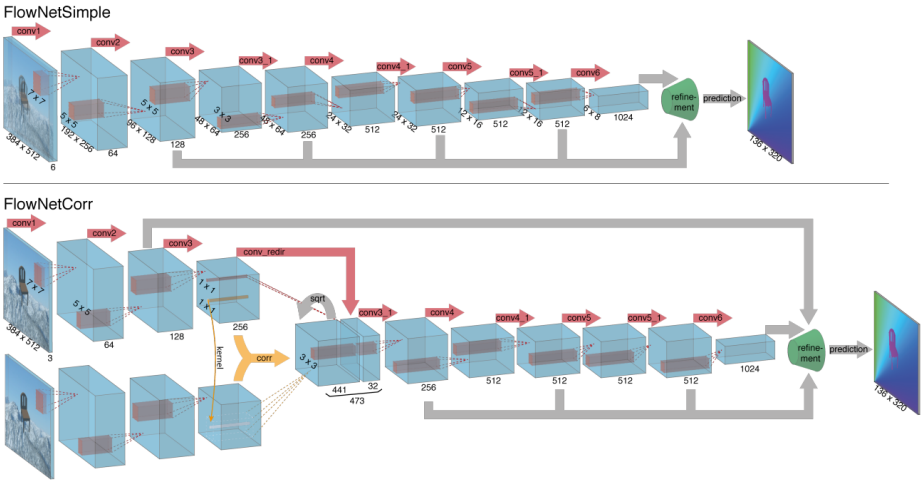

Figura 4: Arquitetura das duas redes propostas no trabalho: FlowNetSimple no topo e FlowNetCorr embaixo [20].

A estimativa do fluxo óptico é uma tarefa de visão computacional de longa data que procura estimar o movimento pixel a pixel de uma sequência, sendo uma informação muito útil para a solução de diversos problemas. Porém, a ideia de utilizar um modelo CNN genérico para aprender o conceito de fluxo óptico diretamente dos dados foi inovadora e completamente desvinculada dos métodos bem estabelecidos na época, introduzindo o problema no campo do Aprendizado Profundo. A FlowNet, como primeira implementação de uma nova corrente de ideias para o campo de estimativa do fluxo óptico, não fugiu à dificuldade de competir com os algoritmos altamente otimizados já existentes, como pode ser visto na Figura 5, mas serviu ao propósito de trazer à tona uma nova forma de se abordar o problema [31].

Desde a revolução introduzida pela FlowNet, as arquiteturas de modelos de estimativa de fluxo óptico vêm evoluindo nos últimos anos, trazendo resultados mais precisos associados a uma maior eficiência computacional, tais como os algoritmos FlowNet2 [31] e PWC-Net [72]. Recentemente, mais grandes avanços vêm sendo feitos na área, introduzindo temas como a atualização do campo de fluxo por meio de unidades recorrentes e a estimativa de fluxo óptico não-supervisionada [30].

2.5 Networks for Large Pixel Displacement

O modelo PWC-Net [72] é um arquitetura do estado da arte para estimativa de fluxo óptico que possui campos receptivos maiores devido à sua estrutura de pirâmide de características

(a) Par de imagens sobrepostas.

(b) Fluxo Óptico esperado.

(c) Fluxo estimado pelo EpicFlow [65].

(d) Fluxo estimado pela FlowNetSimple.

(e) Fluxo estimado pela FlowNetCorr.

Figura 5: Exemplo de predição do Fluxo Óptico de uma sequência de quadros. Da esquerda pra direita, temos: Imagem resultante do par de quadros sobrepostos; Fluxo óptico esperado para os dados de entrada; e predições de fluxo dos métodos EpicFlow, FlowNetSimple e FlowNetCorr, respectivamente. Juntamente com cada predição, a métrica EPE (*endpoint error*) é mostrada, cuja maior proximidade a 0 representa um resultado mais preciso [20].

de 6 níveis, permitindo-o predizer eficientemente grandes movimentos, tendo sido adotado em diversos métodos de VFI como uma alternativa de estimador de fluxo pré-treinado.

A IM-Net [63] é outro modelo que também adota uma estrutura multiescala para reconhecer grandes deslocamentos de objetos em quadros adjacentes, enquanto que sua cobertura é limitada pelo tamanho dos seus filtros adaptativos. Apesar da utilização de robustas estruturas de pirâmide multiescala, os métodos citados anteriormente pecam em capacidade de adaptação, já que o nível mais grosseiro de cada rede é fixado após o treinamento do modelo, ou seja, a escala de cada nível da pirâmide consiste apenas dos seus próprios parâmetros, não compartilhados com o resto da estrutura.

O modelo RRPN [85] tenta solucionar esse problema ao compartilhar os parâmetros de peso através de diferentes níveis de escala em uma estrutura de pirâmide recorrente flexível, entretanto, infere apenas o quadro intermediário no centro do intervalo, apresentando flexibilidade temporal limitada para utilização em VFI para instâncias de tempo alvo arbitrárias. O modelo XVFI-Net [70] se propõe a superar os outros algoritmos ao apresentar uma estrutura escalável com parâmetros compartilhados para variadas resoluções de imagens de entrada, permitindo a predição de um quadro intermediário em um tempo arbitrário t entre os dois quadros de entrada ao utilizar o fluxo complementar reverso de um modo eficiente [70].

2.6 Pruning-based Model Compression

Compressão de modelo [8, 12] é um tópico particularmente importante para modelos de Redes Neurais Profundas, algoritmos conhecidos por depender de altos custos de armazena-

mento e poder computacional, como visto anteriormente. Em geral, as abordagens para essa técnica podem ser categorizadas em 4 tipos: poda (*pruning*), quantização (*quantization*), destilação de conhecimento (*knowledge distillation*) e *AutoML*.

Um dos métodos estudados neste documento, o CDFI [19], adota a técnica de compressão de modelo baseada em poda devido à simplicidade da abordagem, que busca induzir conexões esparsas na estrutura. No caso, a compressão é utilizada como uma ferramenta para um melhor entendimento da base da arquitetura do algoritmo proposto e facilitar melhorias futuras, utilizando técnicas de poda indutoras de esparsidade baseadas em otimização, que envolvem treinamento com restrições de esparsidade, regularizadores, em três passos: treinar o modelo com regularização L1; reformular uma pequena rede totalmente conectada a partir das estruturas esparsas identificadas em cada camada; e treinar novamente a pequena rede para verificar seu desempenho [19].

2.7 Knowledge Distillation

Uma forma muito simples de melhorar o desempenho da maioria dos algoritmos conhecidos de ML é ao treinar vários modelos diferentes em um mesmo conjunto de dados e coletar como métrica resultante a média do desempenho de suas predições [18]. Infelizmente, esse método não é simples de ser utilizado na prática, visto que a utilização de um completo ensemble de modelos para realizar predições pode ser potencialmente muito computacionalmente caro, de forma a inviabilizar a disponibilização da tecnologia para um maior número de usuários, ainda mais se cada estrutura individual se tratar de uma grande rede neural. Buciluǎ et al. [8] conseguiram demonstrar ser possível comprimir o conhecimento adquirido por uma arquitetura completa de ensemble em um único modelo, abordagem esta que foi aprofundada a partir da criação de diferentes técnicas de compressão, como a destilação de conhecimento (*knowledge distillation*) [27].

O método de destilação de conhecimento busca originalmente transferir o aprendizado de um modelo maior para um mais compacto ao realizar o treinamento do modelo destilado em um conjunto de dados de transferência para o qual o modelo mais complexo produziu uma distribuição suave de respostas esperadas [27]. A arquitetura RIFE [30] utiliza a destilação privilegiada (*privileged distillation*) [47], pertencente ao campo da destilação de conhecimento, para calcular o fluxo óptico intermediário dos dados. Nessa técnica mais especializada, um maior número de dados de entrada e características é fornecido para um modelo separado como "professor" em detrimento do modelo "aluno", de forma que o "professor" possa fornecer ao "aluno" uma representação mais precisa para guiá-lo para um melhor processo de aprendizado. O método também é relacionado com a *codistillation* [2], variação onde ambos modelos possuem a mesma arquitetura, porém ainda são fornecidos informações distintas durante o processo de treinamento [30].

2.8 Interpolação de Quadros em Vídeos

A Interpolação de Quadros em Vídeos é uma tarefa fundamental no campo de visão computacional que busca sintetizar, dado um par de quadros consecutivos em um vídeo, um quadro intermediário, que represente exatamente a transição média entre estes, como ilus-

trado nos exemplos da Figura 6, ou, de forma mais geral, qualquer tempo arbitrário t entre os quadros de entrada, aumentando a resolução temporal do conteúdo.

(a) Quadros de entrada sobrepostos [19].

(b) Quadro Intermediário médio esperado [19].

(c) Quadros de entrada sobrepostos [4].

(d) Quadro Intermediário médio esperado [4].

Figura 6: Exemplo Interpolação de Quadros em Vídeos. Nas Figuras 6a e 6c, vemos quadros consecutivos sobrepostos, de dois vídeos diferentes. O objetivo da tarefa então é gerar, nesse caso, o quadro intermediário médio que representa a transição entre os quadros sobrepostos. As respostas esperadas são vistas em 6b e 6d. Nas imagens 6a e 6b, algumas regiões da imagem são mostradas em detalhes para ilustrar melhor o processo [4, 19].

Com o grande sucesso das CNNs em diversos problemas de visão computacional e processamento de imagens, várias técnicas de interpolação de quadros baseadas em Aprendizado Profundo vêm sido desenvolvidas. Long et al. [46] foram pioneiros na utilização de tais métodos no campo, desenvolvendo e treinando uma arquitetura genérica de rede neural convolucional que gera diretamente o quadro intermediário de tempo médio a partir dos quadros consecutivos fornecidos como entrada, baseando-se apenas na capacidade de aprendizado do algoritmo, sem modelar elementos característicos da transição como o fluxo óptico da cena. Porém, o modelo obteve resultados muito desfocados e turvos, devido à falta de capacidade de uma CNN simples de compreender as relações e distribuições complexas atreladas a imagens e vídeos naturais [4, 60].

Os métodos atualmente desenvolvidos para a tarefa podem ser classificados em três grandes categorias, a depender da abordagem utilizada: baseados em fase (*phase-based*), baseados em *kernel* (*kernel-based*) e baseados em fluxo (*flow-based*) [50].

Os algoritmos baseados em fase são aqueles que se valem da mudança de fase de pixels individuais nas imagens para representar o movimento na cena. Tais métodos interpolam os elementos de fase e amplitude através dos níveis de uma pirâmide multiescala utilizando técnicas de otimização [53] ou redes neurais [55], utilizando tais informações para aprender a relação de movimento pertinente aos quadros. Porém, uma desvantagem dessas abordagens é que são aplicáveis apenas para movimentos em um alcance limitado, então, mesmo originando resultados robustos em casos como mudança de iluminação, falham em reconstruir fielmente a textura da imagem [30, 50, 60].

Os métodos baseados em *kernel* são responsáveis por executar os processos de estimativa de movimento e compensação de movimento em uma única etapa [50]. Niklaus et al. [58], com a arquitetura AdaConv [59], estimaram um *kernel* de convolução espacialmente adaptável para cada pixel, usando uma rede convolucional, de forma que o quadro intermediário era gerado ao convoluir os quadros de entrada com os *kernels* preditos. O algoritmo foi futuramente aprimorado utilizando convoluções adaptáveis separáveis, na SepConv, visando diminuir o número de parâmetros da rede e aumentar sua eficiência. Outras abordagens incluem a utilização de fluxo óptico em conjunto com *kernels* de interpolação [4, 5] para melhorar o desempenho e o desenvolvimento de funções de erro que combinam convolução adaptável com interpolação trilinear [63]. Contudo, esses métodos tendem a fornecer resultados borrados ao lidar com objetos em rápido movimento e não conseguem lidar bem com grandes movimentos, maiores que o tamanho estabelecido para o *kernel*, sendo necessário nesse caso aumentar o tamanho dos *kernels*, levando a um aumento exponencial no número de parâmetros da arquitetura [50].

Recentemente, o fluxo óptico tem sido um componente predominante nas abordagens de VFI, sendo estimado nos métodos baseados em fluxo e utilizado na interpolação do quadro intermediário para distorcer os quadros de entrada, explorando explicitamente a informação de movimento da cena [30, 50, 60]. Liu et al. [45] introduziram ao campo de estudo uma rede profunda capaz de produzir vetores tridimensionais de fluxo óptico através do tempo e espaço, distorcendo os quadros de entrada através de amostragem trilinear. [34], por sua vez, propuseram a arquitetura Super-SloMo utilizando a combinação linear dos fluxos ópticos bi-direcionais entre as imagens de entrada como uma aproximação intermediária do fluxo óptico intermediário. Trabalhos recentes na área vêm explorando algumas estratégias

para aumentar o desempenho das abordagens baseadas em fluxo, tais como a utilização de informações contextuais adicionais para interpolar resultados de alta qualidade [56], o desenvolvimento de técnicas não-supervisionadas através da consistência de ciclo [64], a detecção de oclusão na cena ao utilizar a informação de profundidade [4], a distorção para frente (*forward wrapping*) dos quadros de entradas utilizando *softmax splatting* [57], o uso de interpolação quadrática para superar as limitações de modelos lineares [43, 79], o aproveitamento do custo de destilação para supervisionar os fluxos intermediários [30] e a construção de arquiteturas eficientes para o processamento de imagens de alta resolução [16, 70]. No entanto, é importante atentar-se ao fato de que métodos baseados em redes convolucionais geralmente enfrentam desafios ao modelar dependências de longo prazo, limitando a capacidade de lidar com grandes movimentos [50].

Um gráfico comparativo de desempenho de algumas das arquiteturas supracitadas pode ser observado na Figura 7.

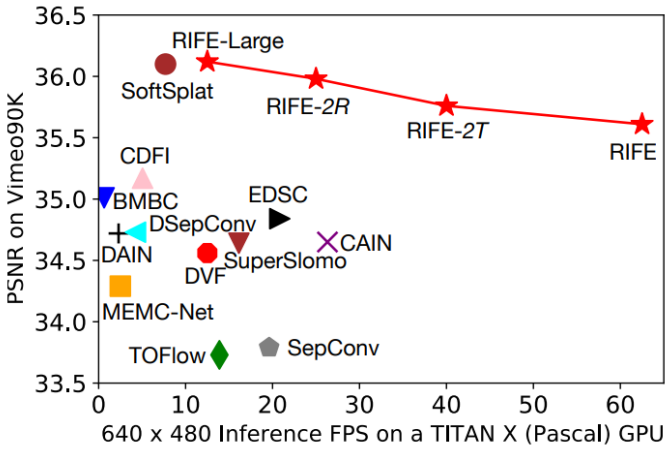

Figura 7: No gráfico, pode-se observar o desempenho e velocidade de alguns modelos recentes propostos para a tarefa de VFI, identificados por símbolos acompanhados de legendas com os respectivos nomes. No eixo vertical, tem-se a precisão média dos algoritmos para a métrica PSNR (*Peak Signal-to-Noise Ratio*) no conjunto de dados Vimeo90K [81]. No eixo horizontal, é dado o número de quadros de resolução 640×480 pixels que cada modelo consegue interpolar por segundo em uma placa de vídeo Titan X da arquitetura Pascal [30].

3 Metodologia

Nesta seção, serão explicitados os passos empregados para execução deste projeto, em ordem, como ilustrado no diagrama da Figura 8. Portanto, na primeira subseção, será explorado o processo de aquisição do conjunto de dados utilizado durante o andamento do trabalho, a segunda subseção explicará por quais métodos de pré-processamento esses dados foram submetidos, a terceira subseção tratará de apresentar as arquiteturas dos métodos estudados neste projeto e o processo de treinamento utilizado e, finalmente, a quarta subseção

abordará os critérios de avaliação adotados para avaliar e comparar o desempenho das três abordagens.

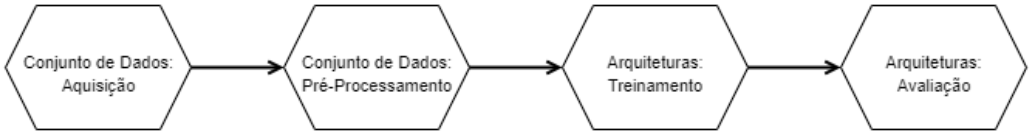

Figura 8: Diagrama ilustrativo das etapas adotadas para a execução deste projeto.

3.1 Conjunto de Dados: Aquisição

A primeira etapa de desenvolvimento do projeto se referiu à sondagem e aquisição de um conjunto de dados adequado ao estudo realizado. Para o treinamento de arquiteturas voltadas à tarefa de Interpolação de Quadros em Vídeos, usualmente são utilizados dados organizados no formato chamado de tripla: um conjunto de três imagens que correspondem a quadros consecutivos retirados de um videoclipe, de tal forma que, seguindo a abordagem do aprendizado supervisionado, o primeiro e o último quadro são fornecidos ao modelo como entrada e o quadro do meio é tido como a saída esperada, já que, de fato, é uma representação perfeita de um quadro intermediário dada a transição entre os outros dois.

Dentre as bases de dados disponíveis neste contexto, a Vimeo90K [81] foi escolhida para a execução do trabalho proposto. O conjunto é o mais popular utilizado para o desenvolvimento de pesquisas na área e se difere dos demais por apresentar imagens de boa qualidade que representam as mais variadas situações, tanto de cenas internas quanto externas, sem a interferência de sinais artificiais introduzidos por *codecs* de vídeo, como ilustrado na Figura 9. Utilizando algoritmos de detecção e ajuste, Xue et al. [81] foram capazes de separar cada vídeo selecionado em capturas consistentes e com cenas de fundo semelhantes, gerando a base de dados completa com um total de 4278 vídeos, dos quais 89800 são imagens independentes, diferindo de contexto umas das outras, sendo redimensionados para a resolução fixa de 448×256 pixels para fins de padronização.

A partir desta base, foi separado um conjunto menor de dados especificamente voltados para a tarefa de VFI, composto de 73171 triplas de quadros retiradas de 14777 videoclipes, escolhidos seguindo 3 critérios: mais de 5% dos pixels das imagens deveriam se movimentar uma distância de pelo menos 3 pixels entre quadros consecutivos, de forma a eliminar vídeos estáticos; os quadros não poderiam possuir mudanças muito grandes em intensidade, já que essa situação não é bem manejada no processo de interpolação; e a diferença média entre campos de movimento de quadros consecutivos deveria ser menor que 1 pixel, removendo casos de movimento não-linear, dado que a maioria dos modelos se baseiam na assunção de linearidade desse elemento [81].

Cada um dos quadros individuais é fornecido como uma imagem no formato *png* e a base de dados é dividida em uma hierarquia de diretórios que coloca cada tripla em uma pasta diferente e as separa a partir das sequências das quais foram respectivamente retiradas. Na Figura 10, pode-se observar alguns exemplos das triplas que populam o

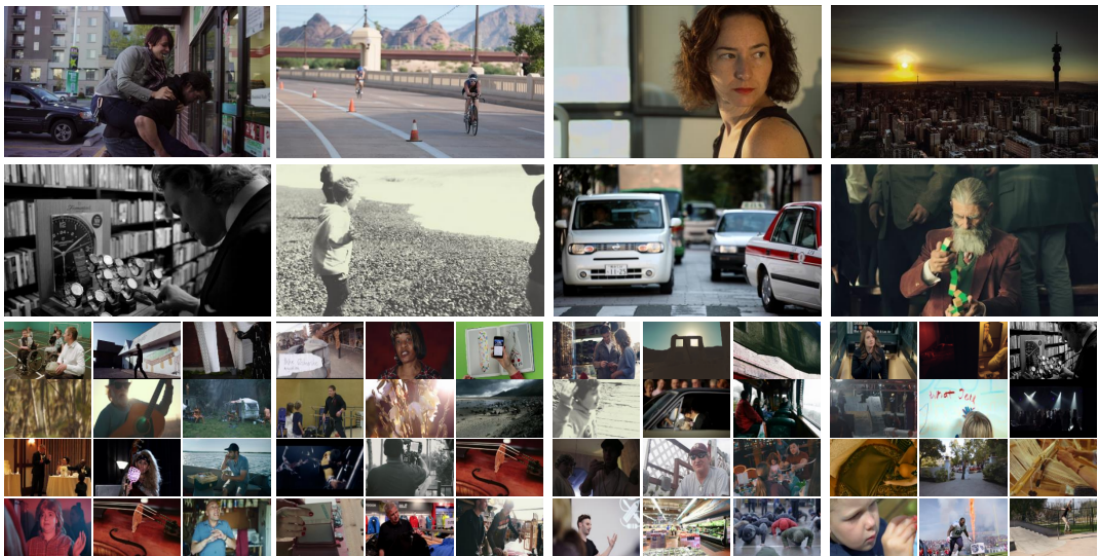

Figura 9: Uma amostra de quadros pertencentes à base de dados Vimeo90K, ilustrando a alta qualidade e variedade de situações das imagens do conjunto [81].

conjunto. Todas as imagens foram originalmente retiradas de vídeos disponibilizados na plataforma de compartilhamento de mídia vimeo.com.

3.2 Conjunto de Dados: Pré-Processamento

Nesta segunda etapa, foi necessário preparar previamente o material adquirido para utilização nos experimentos do trabalho. O conjunto de dados da base Vimeo90K [81], referenciado para uso na tarefa de VFI, possui uma proposta pronta para a divisão dos seus dados em conjuntos de treinamento e teste, especificada em dois arquivos de texto anexados junto aos dados que fornecem a identificação e caminho das amostras dentro da organização no diretório. O registro *tri_trainlist* informa a numeração das 51313 triplas sugeridas para utilização durante o processo de treinamento e otimização de parâmetros dos modelos, enquanto o arquivo *tri_testlist* lista as 3782 amostras, diferentes das anteriores, separadas para serem posteriormente utilizadas durante o processo de avaliação de desempenho dos métodos treinados, em dados novos com padrões ainda não observados pelos algoritmos.

Porém, testes preliminares do projeto demonstraram a inviabilidade da realização de experimentos com a totalidade do conjunto de dados disponível para o processo de treinamento, devido à limitada disponibilidade de poder computacional. Dessa forma, foram selecionadas de forma aleatória, dentre as 51313 amostras sugeridas, 2000 triplas para compor o conjunto de dados de treinamento do escopo do projeto e 400 para a etapa de validação, de forma a avaliar previamente o desempenho dos modelos a cada iteração do processo de treinamento. Essa escolha, embora necessária neste contexto, influencia diretamente no desempenho dos algoritmos utilizados, já que, como visto anteriormente, técnicas baseadas em Aprendizado Profundo possuem seu potencial de aprendizado fortemente vinculado à

(a) Primeiro quadro da sequência. (b) Último quadro da sequência. (c) Primeiro e Último quadros sobrepostos. (d) Quadro intermediário da sequência.

Figura 10: Alguns exemplos de triplas da base de dados Vimeo90K. Na primeira e segunda colunas, pode-se ver os primeiros e últimos quadros, respectivamente, de cada uma das três sequências mostradas. Na terceira coluna, são mostrados o primeiro e o último quadros de cada sequência sobrepostos para efeito de comparação com o quadro intermediário de sequência, na última coluna [81].

quantidade e qualidade das amostras do conjunto de dados disponível para sua análise. Este reflexo será avaliado posteriormente com resultados quantitativos em detalhes na Seção 5.

Ademais, foram aplicadas aos dados uma série de técnicas de aumentação de dados (*data augmentation*), tais como transformações geométricas aleatórias, utilizadas comumente com algoritmos de redes neurais a fim de aumentar o tamanho e qualidade do conjunto de dados de treinamento associado, de forma a otimizar o treinamento e gerar um modelo com maior capacidade de generalização [69]. Para os modelos estudados neste trabalho, os métodos utilizados foram corte (*cropping*), rotação (*rotation*), translação (*translation*), inversão (*flipping*) e reversão na ordem temporal (trocando a posição da primeira e última imagens de uma tripla). Em mais detalhes, na arquitetura RIFE [30] as imagens foram cortadas em pedaços de 224×224 pixels, na XVFI [70] em quadrados de 256×256 pixels e, no modelo CDFI [19], os quadros foram utilizados em sua resolução nativa.

3.3 Arquiteturas: Treinamento

A terceira etapa consiste na efetiva execução dos algoritmos escolhidos, contemplando o processo de treinamento dos modelos, fornecendo-lhes o conjunto de dados selecionado para que aprendam, de fato, a solucionar a tarefa proposta de Interpolação de Quadros em Vídeos. Assim, nesta seção, serão descritas em detalhe as arquiteturas utilizadas e o processo de

treinamento ao qual foram submetidas a fim de aprender os padrões e otimizar os parâmetros necessários para conseguir gerar o quadro intermediário a partir de um par qualquer de imagens de entrada.

De forma geral, a tarefa de VFI é definida para os modelos propostos como a seguir. Dados dois quadros consecutivos, I_0 e I_1, em uma sequência de vídeo, o objetivo da Interpolação de Quadros em Vídeos é sintetizar um quadro intermediário I_t, de posição temporal arbitrária $t \in (0, 1)$. O mais usual é adotar $t = 0.5$, ou seja, gerar o quadro médio entre I_0 e I_1 [19].

3.3.1 CDFI

A ideia da estrutura do CDFI foi proposta como uma instância da arquitetura AdaCof [42], que pode ser identificada principalmente a partir de um de seus principais componentes, uma operação DConv separável espacialmente adaptável que busca sintetizar uma imagem (denotada por I_{out}) a partir de outra (denotada como I_{in}), preenchendo a imagem de entrada de forma a preservar suas dimensões na imagem de saída. A função AdaCof então calcula $I_{out}(i, j)$ ao convoluir um *kernel* deformável de tamanho F ao redor de cada pixel de referência de posição (i, j) em I_{in}, como expresso na função a seguir:

$$\sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{i,j}^{(k,l)} I_{in}(i + dk + \alpha_{i,j}^{(k,l)}, j + dl + \beta_{i,j}^{(k,l)}) \tag{4}$$

Dessa forma, no primeiro passo da abordagem do CDFI, o AdaCof utilizado como modelo base é comprimido ao aproveitar a poda de granularidade fina [89] por meio da otimização indutora de esparsidade [19]. Mais especificamente, dado um modelo completo pré-treinado M_0, seus pesos θ são afinados ao impor uma regularizador de esparsidade L_1, resolvendo o seguinte problema de otimização:

$$\min_{\theta} f(\theta | M_0) + \lambda \left\| \theta \right\|_1, \tag{5}$$

em que $f(.)$ denota o objetivo do treinamento para a tarefa e $\lambda > 0$ a constante de regularização. É conhecido que, ao escolher um lambda apropriado, a formulação (5) promove uma solução esparsa na qual são facilmente identificadas as conexões importantes entre neurônios, correspondentes aos pesos não nulos. Ao resolver o problema regularizado em L_1 (5), a poda de granularidade fina é, de fato, consumada, dado que os zeros são promovidos de forma não-estruturada. Nesse caso é utilizada somente a limitação L_1 devido à sua simplicidade [19].

Após obter uma solução esparsa $\widehat{\theta}$, é reestruturada uma pequena rede densa M_1 baseada na esparsidade computada em cada camada, de forma que o número de *kernels* em cada camada convolucional é atualizado de acordo com a razão de densidade associada, indo de trás para frente. Como o modelo AdaCof é totalmente convolucional (*fully convolutional*), como pode ser visto na Figura 11 que ilustra o desenho da estrutura CDFI, o procedimento idealizado é facilmente implementado pela redução do número de canais de entrada e saída em cada camada, levando a uma arquitetura muito mais compacta [19]. Por fim,

o modelo compactado M_1 é treinado do zero para verificar seu desempenho, processo este que tipicamente leva um tempo significativamente menor devido ao tamanho reduzido da estrutura.

Dessa forma, a estratégia de compressão é então aplicada ao modelo AdaCof pré-treinado fornecido por Lee et al. [42], concluindo durante os testes que a versão comprimida em 10x ainda mantém um desempenho similar à arquitetura original, provando a hipótese de uma considerável redundância no design original [19].

Para o segundo passo, a ideia é melhorar a compressão ao atacar seus pontos de deficiência, dado que é muito mais fácil de operar e testar melhorias no modelo compacto devido aos tempos reduzidos de treinamento e validação. Ao observar problemas do modelo AdaCof em lidar com situações de severa oclusão e preservar pequenos detalhes da cena, Ding et al. [19] propuseram três componentes específicos a serem adicionados à arquitetura AdaCof comprimida: uma pirâmide de características (*feature pyramid*), uma rede de síntese de imagem (*image synthesis network*) e um mecanismo de seleção de caminho (*path selection*).

Na AdaCof [42], o quadro interpolado é calculado ao misturar os dois quadros deformados através de uma única máscara sigmoide V_1 (como pode ser visto na Figura 11), uma generalização da utilização de uma máscara binária para determinar os pesos de oclusão dos dois quadros deformados para cada pixel da saída. [19] argumentaram que a perda de detalhes do contexto nos quadros de entrada é inevitável ao utilizar somente informações brutas dos pixels, por faltar orientação do espaço de características. É extraída então uma representação em pirâmide de características dos quadros de entrada a partir da parte *encoder* da rede U-Net [66] utilizada na AdaCof, possuindo 5 níveis de características, em concordância com o encoder, utilizando uma convolução 1 a 1 em cada nível para filtrar o *encoder* em multiescala com 4, 8, 12, 16 e 20 características de saída, que são finalmente deformadas pela operação AdaCof [42] (Equação 4), que capta movimento no espaço de características.

Para fazer um melhor uso das características multiescala extraídas na estrutura supracitada, Ding et al. [19] recorreram a uma arquitetura GridNet [22] de três linhas e seis colunas para sintetizar a imagem, devido à sua boa capacidade em combinar informação em multiescala. Dessa forma, essa rede de síntese é alimentado com os mapas de característica multiescala deformados tanto para frente quanto para trás, gerando uma única imagem RGB que foca nos detalhes do contexto da cena [19].

Para conseguir aproveitar tanto o modelo AdaCof tratando de movimentos complexos quanto seus próprios componentes lidando com detalhes contextuais, Ding et al. [19] aplicaram um mecanismo de seleção de caminho no processo de gerar o resultado final da interpolação. É aprendido um módulo de oclusão V_2 para sintetizar o resultado final, a partir das saídas da AdaCof original e da rede de síntese, como visto na Figura 11, de forma, na combinação das informações, o resultado da rede de síntese compense a falta de informação contextual na saída da AdaCof.

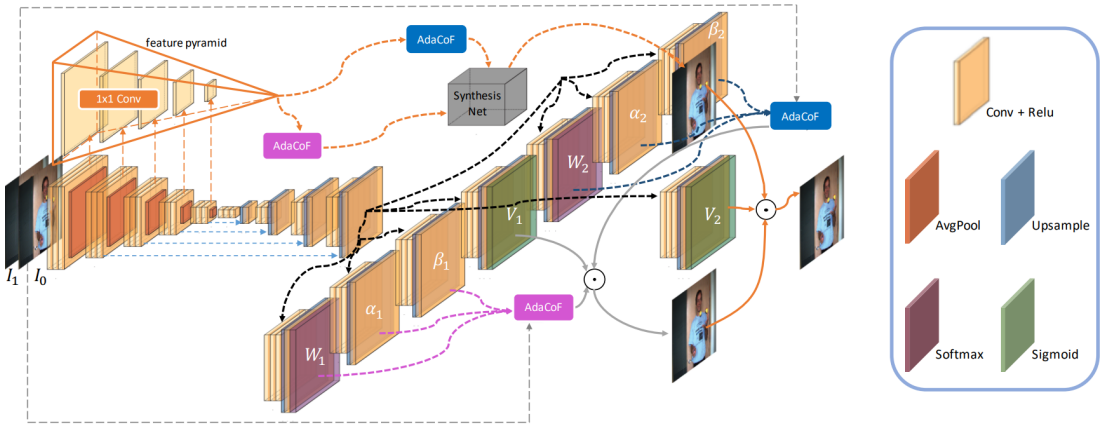

Figura 11: **À esquerda:** Ilustração do projeto de arquitetura do CDFI, baseado no modelo AdaCof [42] comprimido. **À direita:** A legenda mostra os tipos de camadas presentes na arquitetura: em amarelo (*Conv + ReLU*), uma camada convolucional seguida por uma função de ativação ReLU; em laranja (*AvgPool*) uma camada de *Pooling* de média; em azul (*Upsample*), uma camada com um operador de *up-sampling*; em vinho (*Softmax*), uma camada com função de ativação *Softmax*; e, em verde (*Sigmoid*), uma camada com função de ativação *Sigmoid* [19].

3.3.2 RIFE

A arquitetura de RIFE [30], cujo fluxo geral é ilustrado na Figura 12, propõe a estrutura IFNet como modelo estudante no processo de destilação privilegiada para estimar direta e eficientemente o fluxo óptico intermediário a partir dos quadros de entrada I_0 e I_1 e da posição temporal arbitrária t, utilizando uma estratégia *coarse-to-fine* [31] com resolução de aumento gradativo. Os fluxos intermediários $F_{t\to 0}$ e $F_{t\to 1}$ e o mapa de fusão M são estimados ao fornecer à IFNet os quadros de entrada e t como um canal adicional, sintetizando o quadro intermediário $\widehat{I}_t$ a partir da seguinte formulação:

$$\widehat{I}_t = M \odot \widehat{I}_{t\leftarrow 0} + (1 - M) \odot \widehat{I}_{t\leftarrow 1}, \tag{6}$$

$$\widehat{I}_{t\leftarrow 0} = \overleftarrow{W}(I_0, F_{t\to 0}), \quad \widehat{I}_{t\leftarrow 1} = \overleftarrow{W}(I_1, F_{t\to 1}). \tag{7}$$

em que $\overleftarrow{W}$ é a deformação para trás da imagem, $\odot$ é um produto elemento a elemento e M é o mapa de fusão ($0 \leq M \leq 1$). Além disso, Huang et al. [30] utilizaram outra CNN *encoder-decoder*, chamada RefineNet, que possui custo computacional similar à IFNet, para refinar a área de alta frequência da imagem de $\widehat{I}_t$ e reduzir artefatos do modelo estudante no processo de destilação do conhecimento.

A estratégia *coarse-to-fine* com resolução de aumento gradativo, evidenciada na estrutura no formato *stacked hourglass* da IFNet na Figura 13, é utilizada para lidar com o grande movimento presente na tarefa de estimativa de fluxo intermediário, de forma que primeiramente é computada uma predição aproximada do fluxo em baixa resolução, para

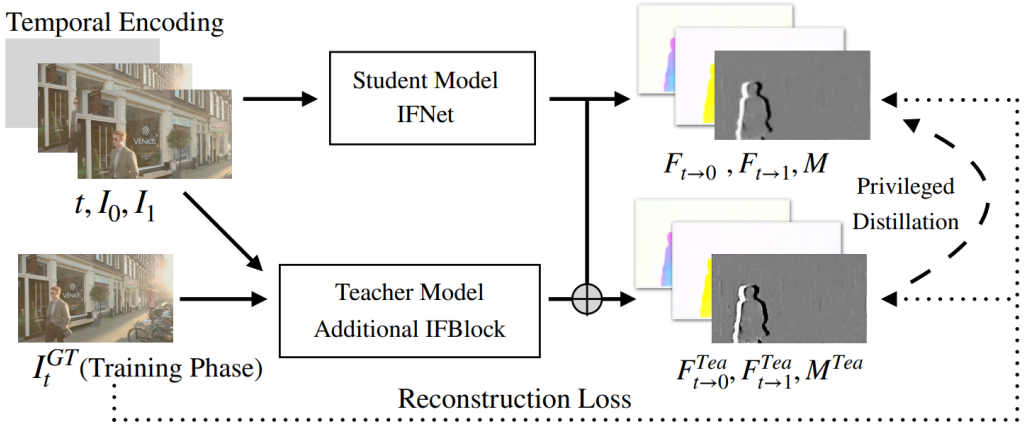

Figura 12: Visão geral da arquitetura RIFE [30].

captar mais facilmente o fator da movimentação, e então os campos de fluxo são iterativamente refinados por meio de sucessivos *IFBlocks*, aumentando gradualmente a resolução do fluxo intermediário estimado:

$$\begin{bmatrix} F^i \\ M^i \end{bmatrix} = \begin{bmatrix} F^{i-1} \\ M^{i-1} \end{bmatrix} + \mathbf{IFB}^i(\begin{bmatrix} F^{i-1} \\ M^{i-1} \end{bmatrix}, t, \widehat{I}^{i-1}), \tag{8}$$

em que F^{i-1} e M^{i-1} denotam as estimativas atuais para os fluxos intermediários e o mapa de fusão do *IFBlock* $(i-1)$ e $\mathbf{IFB}^i$ representa o i-ésimo *IFBlock*. [30] utilizaram um total de 3 IFBlocks com parâmetros de resolução $(K_0, K_1, K_2) = (4, 2, 1)$ e, para manter a simplicidade do projeto cada bloco, possui uma estrutura de *feed-forward* que consiste de algumas camadas convolucionais e um operador de *up-sampling*, utilizando a *PReLU* [26] como a função de ativação para a maioria das camadas.

Huang et al. [30] propuseram também uma função de custo de destilação privilegiada para abordar a dificuldade de estimar fluxos intermediários sem acesso ao quadro intermediário e com falta de supervisão. Dessa forma, é acrescentado ao topo da estrutura um IFBlock que opera como modelo professor ($\mathbf{IFB}^{Tea}$, $K_{Tea} = 1$), responsável por refinar os resultados da IFNet tendo como referência o quadro intermediário esperado I_t^{GT}. Com acesso a I_t^{GT} como informação privilegiada, o modelo professor atinge uma maior precisão na estimativa dos fluxos e a função de custo de destilação calculada a partir disto é utilizada para atualizar o modelo estudante, sem retropropagação deste gradiente para o modelo professor.

3.3.3 XVFI

Em uma arquitetura com um número fixo de níveis de escala, como a PWC-Net [72], dificilmente se adapta a variadas resoluções espaciais do vídeo de entrada, já que a estrutura em cada nível de escala não é compartilhado através dos diferentes níveis, de forma que o processo de treinamento precisaria ser repetido para uma nova arquitetura com maior profundidade de escala. A XVFI-Net [70] é proposta para abordar este problema, possuindo

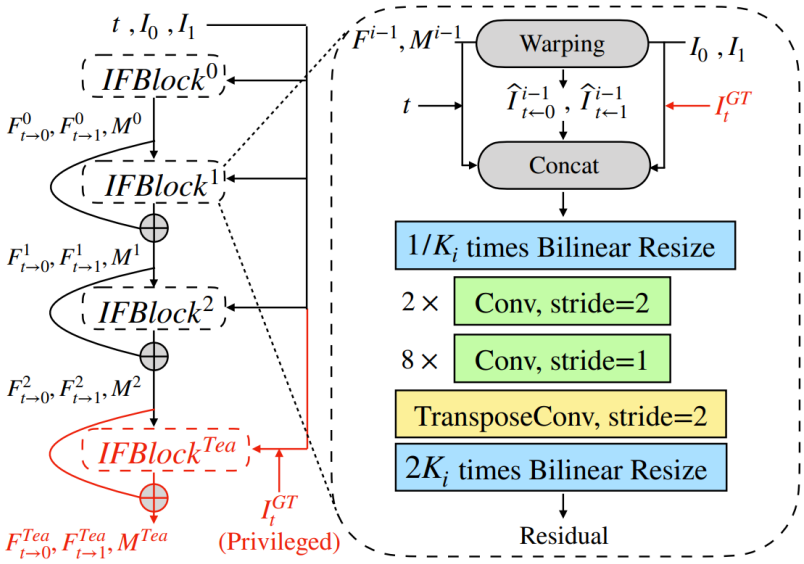

Figura 13: **À esquerda:** A arquitetura da IFNet é composta por vários *IFBlocks* empilhados operando em diferentes resoluções. **À direita:** Dentro de um *IFBlock*, primeiramente são deformados, para frente, os dois quadros de entrada, baseado no fluxo aproximado atual F^{i-1} e então os quadros de entrada I_0 e I_1, os quadros deformados $\widehat{I}_{t\leftarrow 0}$ e $\widehat{I}_{t\leftarrow 1}$, os resultados anteriores F^{i-1} e M^{i-1} e o passo temporal t são alimentados ao próximo bloco para aproximar os resíduos de fluxo e máscara. A informação privilegiada I_t^{GT} somente é fornecida ao modelo professor [30].

uma estimativa de fluxo óptico que possa começar em qualquer nível de escala grosseiro desejado, adaptando-se ao grau de movimentação presente nos quadros de entrada através do compartilhamento de parâmetros entre diferentes níveis de escala.

Para captar grandes movimentos entre os dois quadros de entrada, o bloco de extração de características da XVFI-Net [70], composto por uma camada convolucional com *stride* 2 e dois blocos residuais, como mostrado na Figura 14, primeiramente reduz a resolução espacial das imagens por um fator escalar modular M por meio da sua operação de convolução de forma que as características espacialmente reduzidas sejam então convertidas para dois mapas de características contextuais, C_0^0 e C_1^0. Posteriormente, o modelo estima, em cada nível de escala, o fluxo óptico do quadro intermediário alvo I_t para cada quadro de entrada em tamanho reduzido por M, de forma que os fluxos preditos têm sua resolução aumentada novamente por um fator de M para deformar os quadros de entrada em cada nível de escala para o tempo arbitrário t.

É possível observar na Figura 15 a arquitetura da XVFI-Net [70] para um nível de escala s, composta pelos módulos BiOF-I e BiOF-T. Primeiro, uma pirâmide contextual $\mathbf{C} = C^s$ é extraída de C_0^0 e C_1^0 por meio da convolução de *stride* 2, e então utilizada como entrada para a XVFI-Net em cada nível de escala s $(s = 0, 1, 2, ...)$, em que $s = 0$ representa a escala original dos quadros de entrada. Os mapas de características contextuais junto com os fluxos bidirecionais iniciais entre os quadros de entrada, ambos do nível de escala atual, são

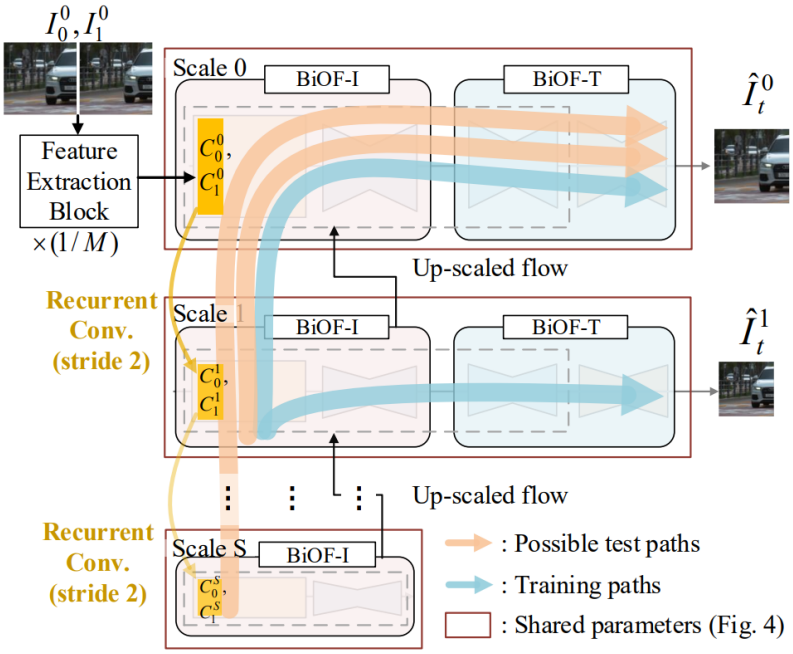

Figura 14: Escalabilidade eficiente e ajustável da estrutura da XVFI-Net [70].

alimentados para a estrutura BiFlownet baseada em *auto-encoder*, ilustrada na Figura 15, que retorna fluxos residuais sobre os fluxos iniciais e uma máscara de importância treinável z. Dessa forma, são obtidos os fluxos bidirecionais entre os quadros de entrada ao nível de escala s, F_{01}^s e F_{10}^s, que são fornecidos como entrada ao módulo BiOF-T e utilizados como fluxos iniciais para o próximo nível de escala $s-1$.

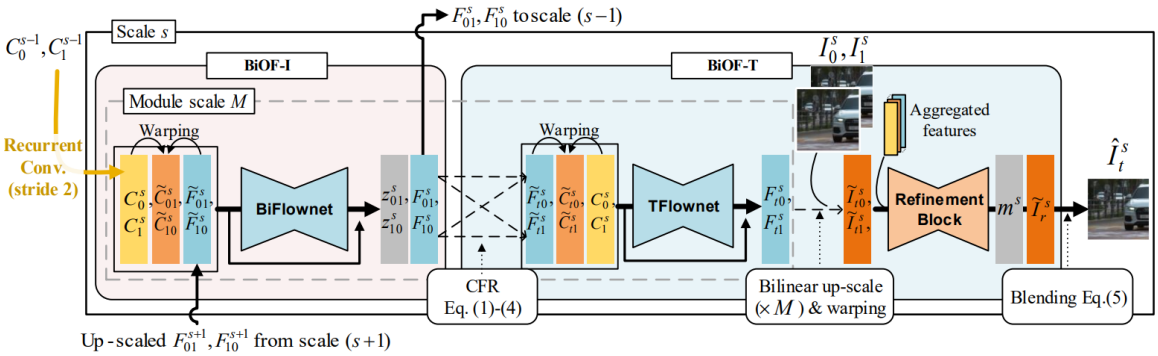

Figura 15: A arquitetura da XVFI-Net em uma escala arbitrária s [70].

No módulo BiOF-T [70], uma estimativa de fluxo óptico estável do tempo t para o tempo 0 ou 1, dos quadros de entrada, pode ser computada através de uma combinação linear normalizada de um *negative anchor flow* e um *complementary flow*, operação chamada de de CFR (*complementary flow reversal*). Os mapas de fluxo óptico revertido comple-

mentar, $\tilde{F}^s_{t0}$ e $\tilde{F}^s_{t1}$, de t para 0 e 1, são então refinados ao deformar novamente os mapas de características e alimentar todas as informações para a estrutura TFlownet baseada em *auto-encoder*, mostrada na Figura 15. Os resultados têm sua resolução aumentada de volta ao tamanho do quadro intermediário, de forma que é criado um agregado com todas as características usadas e fornecido ao bloco de refinamento baseado em U-Net [66], cujas saídas são combinadas para gerar o quadro intermediário resultante final de cada nível de escala. Atualizando o fluxo da arquitetura gradualmente até a escala original $s = 0$ é possível então sintetizar adequadamente o quadro intermediário desejado.

3.3.4 Treinamento dos Modelos

As arquiteturas apresentadas foram treinadas utilizando o conjunto reduzido do Vimeo90K [81] contendo 2000 triplas para treinamento e 400 triplas para validação. Para efetuar o processo foram utilizados os códigos de treinamento disponibilizados pelos autores nos respectivos repositórios oficiais hospedados na plataforma GitHub, com algumas mudanças para adequá-los à estrutura disponibilizada para execução do projeto. Mais detalhes sobre as condições de treinamento e a especificação dos parâmetros utilizados durante o processo serão expostos na Seção 4.

3.4 Arquiteturas: Avaliação

Nesta seção serão descritas as métricas de avaliação utilizadas no projeto para medir o desempenho de cada uma das arquiteturas estudadas. Nesse contexto, foram adotadas as medidas de PSNR (*Peak Signal-to-Noise Ratio*) e de SSIM (*Structural Similarity Index*), por serem as mais comumente utilizadas para apresentar resultados na área de Interpolação de Quadro em Vídeos.

3.4.1 PSNR

A Razão Sinal-Ruído de Pico (PSNR, do inglês *Peak Signal-to-Noise Ratio*) é a medida que expressa a relação entre a máxima energia possível de um sinal e o poder de corromper de um ruído que afeta a qualidade e fidelidade de sua representação. A razão é comumente utilizada na engenharia como medida de qualidade entre uma imagem original e uma versão comprimida ou reconstruída sua. Como muitos sinais possuem uma faixa dinâmica muito larga, o PSNR é normalmente expresso em escala logarítmica, utilizando o decibel (dB) como unidade, e é calculado a partir da seguinte fórmula para um canal (R, G ou B):

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\text{MSE}} \right), \tag{9}$$

sendo MAX o maior valor que um pixel pode admitir na imagem, comumente 255 para pixels representados utilizando 8 bits por amostra, e MSE o Erro Quadrático Médio (*Mean Squared Error*), calculado entre duas imagens I_0 e I_1, de M linhas e N colunas, como:

$$\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left[I_0(i,j) - I_1(i,j) \right]^2. \tag{10}$$

Quanto maior o valor do PSNR, melhor a qualidade da imagem comprimida ou reconstruída em relação à original.

3.4.2 SSIM

O Índice de Similaridade Estrutural (SSIM, do inglês *Structural Similarity Index*) foi proposto por Wang et. al. [75] em 2004 com a ideia de criar abordagem mais direta para comparar as estruturas entre um sinal de referência e um sinal distorcido ou reconstruído a partir do primeiro. Para isso, ele define como a informação estrutural de uma imagem como os atributos que representam a estrutura dos objetos presentes na cena, independentemente da luminância ou contraste médios, já que podem variar muito através do sinal, utilizando então a luminância e contraste locais para a definição.

Considerando duas imagens x e y, alinhadas entre si e cujo uma das amostras representa a noção de qualidade perfeita, como a saída esperada na tarefa de VFI, a medida de similaridade pode ser utilizada como uma métrica quantitativa de qualidade da outra amostra [75]. O cálculo da métrica SSIM é então realizado ao combinar as comparações entre os elementos de luminância, contraste e estrutura dos sinais entre si, segundo a seguinte fórmula:

$$\mathrm{SSIM}(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}, \tag{11}$$

em que $l(x,y)$ é a função de comparação de luminância entre os sinais x e y, $c(x,y)$ é a função de comparação de contraste entre os sinais x e y, $s(x,y)$ é a função de comparação de estrutura entre os sinais x e y e $\alpha > 0$, $\beta > 0$ e $\gamma > 0$ são parâmetros utilizados para ajustar a importância relativa dos três componentes, normalmente escolhidos como $\alpha = \beta = \gamma = 1$ por simplicidade [75]. As funções de comparação, por sua vez, são definidas como a seguir:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \qquad c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \qquad s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{12}$$

em que μ_x, μ_y, σ_x, σ_y e σ_{xy} são as médias locais de intensidade de cada imagem, os desvios padrões de cada imagem e o coeficiente de correlação entre as imagens, respectivamente, e C_1, C_2 e C_3 são constantes inclusas para evitar instabilidade quando os valores do denominador forem muito próximos de 0.

4 Experimentos

Durante os experimentos efetuados no decorrer deste trabalho, foi utilizada primariamente a linguagem de programação Python, com a utilização de algumas ferramentas e bibliotecas auxiliares, entre as quais se destacam a plataforma Jupyter Notebook, como interface, e o pacote PyTorch, utilizado como biblioteca de Aprendizado Profundo para implementação das arquiteturas estudadas e seus procedimentos relacionados. Os códigos para treinamento dos modelos foram adaptados dos repositórios disponibilizados pelos autores [19, 30, 70] e executados nos ambientes de máquina virtual do Google Colaboratory, com uma GPU Nvidia K80 de 12GB, uma CPU Intel Xeon Haswell de dois núcleos e 12GB de memória RAM, e do Kaggle, com uma GPU Nvidia P100 de 16GB, uma CPU Intel Xeon Haswell de

quatro núcleos e 16 GB de memória RAM. Durante o processo, os conjuntos de treinamento, validação e teste foram utilizados, explicitados na Seção 3.2, contendo 2000, 400 e 3782 triplas, respectivamente, extraídas da base de dados Vimeo90K [81].

A primeira arquitetura a passar pelo processo de treinamento dos seus parâmetros foi a RIFE [30], utilizando o otimizador AdamW [48] com decaimento de pesos de 1×10^{-4}, um *batch_size* de 32 e uma taxa de aprendizado começando em 3×10^{-4} e sendo gradualmente diminuída a 3×10^{-5} por meio de *cosine annealing* durante todo o treinamento de 100 épocas de duração, realizado na plataforma do Google Colab. O modelo XVFI-Net [70] foi treinado com o otimizador Adam [37], taxa de aprendizado de 1×10^{-4} e um *batch_size* de 8 no Kaggle, durante 100 épocas. Por sua vez, o CDFI [19] utilizou em seu processo de treinamento o algoritmo de otimização AdaMax [37], juntamente com uma taxa de aprendizado inicial de 1×10^{-3}, decaindo pela metade a cada 20 das 100 épocas do treinamento e um *batch_size* de 4. Todos os otimizadores foram configurados com os valores padrão de $\beta_1 = 0.9$ e $\beta_2 = 0.999$.

Os valores de *batch_size* para cada um dos modelos durante o processo de treinamento foi reduzido dos valores sugeridos de 64, 16 e 8 para as arquiteturas RIFE, XVFI e CDFI, respectivamente, alterando os parâmetros sugeridos nos artigos [19, 30, 70] devido à limitação de poder computacional das placas de vídeo disponibilizadas pelas plataformas utilizadas, incapazes de concluir o processo com os valores originais. Além disso, o número de épocas dos algoritmos RIFE e XVFI foi reduzido dos valores recomendados de 300 e 200 [30, 70], respectivamente, para 100 por causa do tempo máximo de execução disponibilizado pelas plataformas para uma sessão de treinamento.

5 Resultados e Discussão

Após finalizado o processo de treinamento, conforme descrito na Seção 4, cuja execução requereu aproximadamente 10 horas para cada arquitetura, os modelos RIFE, XVFI e CDFI tiveram seu desempenho avaliado no conjunto separado para teste da base de dados Vimeo90K [81], assim como especificado na Seção 3.2. O desempenho das redes foi medido para as duas métricas selecionadas para o trabalho e apresentadas na Seção 3.4, PSNR e SSIM, e os resultados obtidos podem ser observados na Tabela 1.

Tabela 1: Resultados quantitativos dos modelos treinados no conjunto de teste do conjunto Vimeo90K [81]. Os valores em **vermelho** representam os melhores desempenhos obtidos em cada métrica.

Arquitetura	PSNR	SSIM
CDFI	33.24	0.941
RIFE	33.37	**0.964**
XVFI	**33.51**	0.953

Comparando os valores obtidos com o desempenho apresentado pelos modelos em seus respectivos trabalhos de origem, tal como ilustrado na Tabela 2, pode-se notar que, embora os resultados sejam um pouco piores, as arquiteturas ainda conseguiram efetuar um

bom processo de aprendizado e convergir para obter resultados consideráveis mesmo tendo disponíveis menos de 5% dos dados originais e parâmetros reduzidos, demonstrando a robustez e consistência dos algoritmos estudados neste trabalho e o sucesso das suas diferentes abordagens para captar as informações da cena na tarefa de VFI. Ademais, a redução do desempenho é esperada devido à redução no tamanho do conjunto de dados e no número de épocas de treinamento, visto que dessa forma os modelos possuem menos dados a partir dos quais aprender os padrões, já que em 1 época com o conjunto original os algoritmos eram apresentados a um número 25 vezes maior de informações em relação aos experimentos deste projeto.

Tabela 2: Resultados dos modelos reportados em seus respectivos artigos [19, 30, 70]. Os valores em **vermelho** representam os melhores desempenhos obtidos em cada métrica.

Arquitetura	PSNR	SSIM
CDFI	35.17	0.964
RIFE	**35.61**	**0.978**
XVFI	35.07	0.976

Para avaliar o desempenho destas arquiteturas as duas métricas devem ser observadas em conjunto, já que avaliam a comparação da imagem reconstruída com a original em diferentes âmbitos, igualmente relevantes e complementares para obter um resultado consistente. Pode-se notar que, no ambiente criado para o projeto, o modelo XVFI obteve um melhor resultado para a métrica de PSNR, enquanto RIFE alcançou o melhor valor para a métrica de SSIM, entretanto, as diferenças observadas foram muito pequenas e, no geral, todos os modelos apresentaram bons resultados com valores equilibrados comparando as métricas.

No entanto, os valores das métricas calculados a partir do desempenho médio dos algoritmos no conjunto de teste, embora semelhantes, podem refletir comportamentos muito diferentes na prática ao gerar os quadros intermediários, evidenciando a capacidade das arquiteturas de reconhecerem melhor padrões em algumas situações do que em outras a partir do que aprenderam no processo de treinamento. Por exemplo, a RIFE poderia apresentar ótimo desempenho em uma imagem e o XVFI criar um quadro de péssima qualidade para a situação e em uma outra sequência o XVFI gerar uma imagem impecável e a RIFE falhar em interpretar as informações necessárias para interpolar os detalhes do quadro médio.

Como toda aplicação prática de AI, é essencial também avaliar o desempenho qualitativo do experimento. É possível ver nas Figuras 16, 17 e 18 que os três modelos conseguiram estimar com qualidade satisfatória o quadro médio para 3 triplas escolhidas aleatoriamente do conjunto de teste, apresentando apenas algumas imperfeições e falta de detalhes que muito provavelmente poderiam passar despercebidas por um usuário na exibição da sequência completa. Com as condições ótimas de treinamento para os modelos (como no desempenho evidenciado pelas métricas da Tabela 2), as quais não foram possíveis de serem reproduzidas no escopo deste trabalho, a qualidade da imagem gerada também aumenta consideravelmente, com detalhes mais claros.

(a) Primeiro quadro de entrada.

(b) Segundo quadro de entrada.

(c) Quadros de entrada sobrepostos.

(d) Resposta esperada, quadro intermediário da tripla.

(e) Quadro intermediário gerado pelo RIFE [30].

Figura 16: Exemplo de resultado qualitativo da arquitetura RIFE [30] ao interpolar o quadro médio para uma tripla pertencente ao conjunto de teste da base Vimeo90K [81].

6 Conclusões

A execução deste projeto possibilitou uma exploração mais minuciosa de uma área de estudo muito relevante dentro da Computação Visual e com muitas aplicações práticas, entretanto, pouco conhecida pelo público geral, a Interpolação de Quadros em Vídeos, cuja alta complexidade está muito associada aos elementos cotidianos facilmente capturados em mídia, mas de difícil representação e interpretação por modelos matemáticos, refletidos no conjunto de dados utilizado, Vimeo90K [81]. Como parte do trabalho, três arquiteturas do estado da arte com diferentes abordagens para abordar o problema em questão foram estudadas, utilizando algoritmos de Aprendizado Profundo para capturar informações a respeito da cena de cada imagem.

O modelo CDFI [19] explorou a compressão de modelos já bem estabelecidos e a utilização de técnicas de pirâmide de características e seleção de caminhos para melhorar o desempenho e a eficiência computacional, enquanto a arquitetura RIFE [30] partiu do conceito de destilação de conhecimento para treinar um módulo com informações privilegiadas para guiar um treinamento efetivo do modelo base mais eficiente para aprender a representar corretamente o fluxo óptico intermediário da cena retratada pelos quadros de entrada, e o algoritmo da XVFI [70] criou uma estrutura multiescala para estimar o fluxo óptico e os mapas de características associados ao contexto das imagens em diferentes níveis de forma eficiente e escalável, com retroalimentação das representações geradas e dos padrões aprendidos.

(a) Primeiro quadro de entrada.

(b) Segundo quadro de entrada.

(c) Quadros de entrada sobrepostos.

(d) Resposta esperada, quadro intermediário da tripla.

(e) Quadro intermediário gerado pelo CDFI [19].

Figura 17: Exemplo de resultado qualitativo da arquitetura CDFI [19] ao interpolar o quadro médio para uma tripla pertencente ao conjunto de teste da base Vimeo90K [81].

Foi possível observar o alto nível de consistência das propostas dos modelos estudados, visto que todos alcançaram resultados quantitativos superiores a 33 dB de PSNR (*Peak Signal-to-Noise Ratio*) e 0.940 de SSIM (*Structural Similarity Index*), conseguindo alcançar convergência nos processos de treinamento e identificar os elementos necessários à interpolação da cena mesmo trabalhando com um conjunto reduzido de informações. No contexto do projeto, as arquiteturas XVFI e RIFE foram as que melhor conseguiram se adequar à situação adversa e extrair os padrões necessários dos poucos dados disponíveis, apresentando os maiores valores para métricas avaliadas: 33.51 dB de PSNR e 0.964 de SSIM, respectivamente. Ademais, embora tenham apresentado um resultado inferior ao nominal registrado em seus respectivos artigos, qualitativamente todos os modelos se mostraram capazes de gerar quadros com qualidade satisfatória a fim de aumentar a taxa de quadros de um vídeo sem uma grande perda de detalhes perceptível na média.

A maior dificuldade enfrentada no trabalho se relaciona diretamente ao fato de os algoritmos estudados fazerem parte do estado da arte nesta área de estudo, necessitando de elevado poder computacional para treinar seus modelos e executar suas propostas de modo eficiente, o qual não está amplamente disponível e não pôde ser alcançado no escopo do projeto. Mesmo com as aplicações prontas podendo ser bem executadas utilizando apenas CPUs, o difícil acesso ao processo de treinamento desestimula a pesquisa e retarda o avanço do campo. Além disso, como abordado por Kim Sim et al. [70], o desenvolvimento de abordagens voltadas a vídeos de alta resolução ainda é muito incipiente. O conjunto de dados Vimeo90K [81], embora represente com muita qualidade e variedade diversas situações do

(a) Primeiro quadro de entrada.

(b) Segundo quadro de entrada.

(c) Quadros de entrada sobrepostos.

(d) Resposta esperada, quadro intermediário da tripla.

(e) Quadro intermediário gerado pelo XVFI [70].

Figura 18: Exemplo de resultado qualitativo da arquitetura XVFI [70] ao interpolar o quadro médio para uma tripla pertencente ao conjunto de teste da base Vimeo90K [81].

cotidiano, possui imagens de baixa resolução, desconexa dos valores praticados no mercado de monitores e televisores, por exemplo.

Refletindo sobre os problemas apontados, algumas sugestões para próximos passos da pesquisa na área de Interpolação de Quadros em Vídeos incluem a busca pela otimização do processo de treinamento e execução dos modelos, principalmente associados a conteúdos de resoluções mais altas, para maior desenvolvimento da área. Ademais, também nota-se relevante testar o uso de novas e inovadoras tecnologias de redes neurais para a tarefa, como Transformers [50], que podem trazer melhoras significativas para o desempenho e eficiência dos programas.

Referências

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2020.

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[3] W. Bao, X. Zhang, L. Chen, L. Ding, and Z. Gao. High-order model and dynamic filtering for frame rate up-conversion. *IEEE Transactions on Image Processing*, 27(8): 3813–3826, 2018.

[4] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang. Depth-aware video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[5] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang. MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 (3):933–948, 2019.

[6] C. M. Bishop and N. M. Nasrabadi. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

[7] B. G. Buchanan. A (very) brief history of artificial intelligence. *AI Magazine*, 26(4): 53–53, 2005.

[8] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541, 2006.

[9] R. Castagno, P. Haavisto, and G. Ramponi. A method for motion adaptive frame rate up-conversion. *IEEE Transactions on circuits and Systems for Video Technology*, 6(5): 436–446, 1996.

[10] Z. Chen, R. Wang, H. Liu, and Y. Wang. PDWN: Pyramid deformable warping network for video interpolation. *IEEE Open Journal of Signal Processing*, 2:413–424, 2021.

[11] X. Cheng and Z. Chen. Video frame interpolation via deformable separable convolution. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 10607–10614, 2020.

[12] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[13] Z. Chi, R. Mohammadi Nasiri, Z. Liu, J. Lu, J. Tang, and K. N. Plataniotis. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In *European Conference on Computer Vision*, pages 107–123. Springer, 2020.

[14] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko. Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(4): 407–416, 2007.

[15] M. Choi, H. Kim, B. Han, N. Xu, and K. M. Lee. Channel attention is all you need for video frame interpolation. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 10663–10671, 2020.

[16] M. Choi, S. Lee, H. Kim, and K. M. Lee. Motion-aware dynamic architecture for efficient frame interpolation. In *IEEE/CVF International Conference on Computer Vision*, pages 13839–13848, 2021.

[17] S. Daly. Engineering observations from spatiovelocity and spatiotemporal visual models. In *Vision Models and Applications to Image and Video Processing*, pages 179–200. Springer, 2001.

[18] T. G. Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[19] T. Ding, L. Liang, Z. Zhu, and I. Zharkov. CDFI: Compression-driven network design for frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8001–8011, 2021.

[20] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[21] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[22] D. Fourure, R. Emonet, E. Fromont, D. Muselet, A. Tremeau, and C. Wolf. Residual conv-deconv grid network for semantic segmentation. *arXiv preprint arXiv:1707.07958*, 2017.

[23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.

[24] S. Gui, C. Wang, Q. Chen, and D. Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14004–14013, 2020.

[25] M. Haris, G. Shakhnarovich, and N. Ukita. Space-time-aware multi-resolution video enhancement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2859–2868, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[27] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[28] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[29] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17 (1-3):185–203, 1981.

[30] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou. RIFE: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2020.

[31] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470, 2017.

[32] T. Jayashankar, P. Moulin, T. Blu, and C. Gilliam. Lap-based video frame interpolation. In *IEEE International Conference on Image Processing*, pages 4195–4199. IEEE, 2019.

[33] S.-G. Jeong, C. Lee, and C.-S. Kim. Motion-compensated frame interpolation based on multihypothesis motion estimation and texture optimization. *IEEE Transactions on Image Processing*, 22(11):4497–4509, 2013.

[34] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super-SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[35] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics*, 35(6):1–10, 2016.

[36] S. Y. Kim, J. Oh, and M. Kim. FISR: Deep joint frame interpolation and super-resolution with a multi-scale temporal loss. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 11278–11286, 2020.

[37] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] M. Kubas and G. Sarwas. Fastrife: Optimization of real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2105.13482*, 2021.

[39] Y. Kuroki, T. Nishi, S. Kobayashi, H. Oyaizu, and S. Yoshimura. A psychophysical study of improvements in motion-image quality by using high frame rates. *Journal of the Society for Information Display*, 15(1):61–68, 2007.

[40] Y. Kuroki, H. Takahashi, M. Kusakabe, and K.-I. Yamakoshi. Effects of motion image stimuli with normal and high frame rates on EEG power spectra: comparison with continuous motion image stimuli. *Journal of the Society for Information Display*, 22 (4):191–198, 2014.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[42] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5316–5325, 2020.

[43] Y. Liu, L. Xie, L. Siyao, W. Sun, Y. Qiao, and C. Dong. Enhanced quadratic video interpolation. In *European Conference on Computer Vision*, pages 41–56. Springer, 2020.

[44] Y.-L. Liu, Y.-T. Liao, Y.-Y. Lin, and Y.-Y. Chuang. Deep video frame interpolation using cyclic frame generation. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 8794–8802, 2019.

[45] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *IEEE International Conference on Computer Vision*, pages 4463–4471, 2017.

[46] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016.

[47] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[48] I. Loshchilov and F. Hutter. Fixing weight decay regularization in Adam. *openreview.net*, pages 1–14, 2018.

[49] G. Lu, X. Zhang, L. Chen, and Z. Gao. Novel integration of frame rate up conversion and HEVC coding based on rate-distortion optimization. *IEEE Transactions on Image Processing*, 27(2):678–691, 2017.

[50] L. Lu, R. Wu, H. Lin, J. Lu, and J. Jia. Video frame interpolation with transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3532–3542, 2022.

[51] A. Mackin, F. Zhang, and D. R. Bull. A study of subjective video quality at various frame rates. In *IEEE International Conference on Image Processing*, pages 3407–3411. IEEE, 2015.

[52] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[53] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1418, 2015.

[54] S. Meyer, V. Cornillère, A. Djelouah, C. Schroers, and M. Gross. Deep video color propagation. *arXiv preprint arXiv:1808.03232*, 2018.

[55] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers. PhaseNet for video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 498–507, 2018.

[56] S. Niklaus and F. Liu. Context-aware synthesis for video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1710, 2018.

[57] S. Niklaus and F. Liu. Softmax splatting for video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020.

[58] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[59] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[60] J. Park, K. Ko, C. Lee, and C.-S. Kim. BMBC: Bilateral motion estimation with bilateral cost volume for video interpolation. In *European Conference on Computer Vision*, pages 109–125. Springer, 2020.

[61] J. Park, C. Lee, and C.-S. Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *IEEE/CVF International Conference on Computer Vision*, pages 14539–14548, 2021.

[62] H. Pedrini and W. R. Schwartz. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. Cengage Learning, 2007.

[63] T. Peleg, P. Szekely, D. Sabo, and O. Sendik. IM-Net for high resolution video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2398–2407, 2019.

[64] F. A. Reda, D. Sun, A. Dundar, M. Shoeybi, G. Liu, K. J. Shih, A. Tao, J. Kautz, and B. Catanzaro. Unsupervised video interpolation using cycle consistency. In *IEEE/CVF International Conference on Computer Vision*, pages 892–900, 2019.

[65] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[66] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[67] W. Shen, W. Bao, G. Zhai, L. Chen, X. Min, and Z. Gao. Blurry video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5114–5123, 2020.

[68] Z. Shi, X. Liu, K. Shi, L. Dai, and J. Chen. Video frame interpolation via generalized deformable convolution. *IEEE Transactions on Multimedia*, 24:426–439, 2021.

[69] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[70] H. Sim, J. Oh, and M. Kim. Xvfi: Extreme video frame interpolation. In *IEEE/CVF International Conference on Computer Vision*, pages 14489–14498, 2021.

[71] L. Siyao, S. Zhao, W. Yu, W. Sun, D. Metaxas, C. C. Loy, and Z. Liu. Deep animation video interpolation in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6587–6595, 2021.

[72] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[73] Y. Tian, Y. Zhang, Y. Fu, and C. Xu. TDAN: Temporally-deformable alignment network for video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3360–3369, 2020.

[74] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy. EDVR: Video restoration with enhanced deformable convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2019.

[75] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[76] C.-Y. Wu, N. Singhal, and P. Krahenbuhl. Video compression through image interpolation. In *European Conference on Computer Vision*, pages 416–431, 2018.

[77] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen. Modeling and optimization of high frame rate video transmission over wireless networks. *IEEE Transactions on Wireless Communications*, 15(4):2713–2726, 2015.

[78] J. Xing, W. Hu, Y. Zhang, and T.-T. Wong. Flow-aware synthesis: A generic motion model for video frame interpolation. *Computational Visual Media*, 7(3):393–405, 2021.

[79] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang. Quadratic video interpolation. *Advances in Neural Information Processing Systems*, 32, 2019.

[80] F. Xue, J. Li, J. Liu, and C. Wu. Bwin: A bilateral warping method for video frame interpolation. In *IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2021.

[81] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[82] S. Yu and J. Jeong. Hierarchical motion estimation algorithm using multiple candidates for frame rate up-conversion. In *International Workshop on Advanced Image Technology*, volume 11049, pages 636–640. SPIE, 2019.

[83] J. Zhai, K. Yu, J. Li, and S. Li. A low complexity motion compensated frame interpolation method. In *IEEE International Symposium on Circuits and Systems*, pages 4927–4930. IEEE, 2005.

[84] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[85] H. Zhang, Y. Zhao, and R. Wang. A flexible recurrent residual pyramid network for video frame interpolation. In *European Conference on Computer Vision*, pages 474–491. Springer, 2020.

[86] B. Zhao and X. Li. Ea-net: Edge-aware network for flow-based video frame interpolation. *arXiv preprint arXiv:2105.07673*, 2021.

[87] K. Zhou, W. Li, X. Han, and J. Lu. Exploring motion ambiguity and alignment for high-quality video frame interpolation. *arXiv preprint arXiv:2203.10291*, 2022.

[88] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016.

[89] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.