Feature Distillation in Video Frame Interpolation

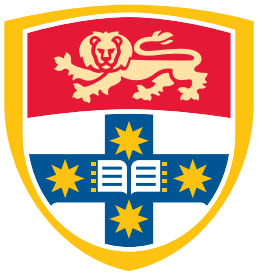
XIAOXUAN LI
SID: 500703140

Supervisor: Dr. Wei Bao

This thesis is submitted in partial fulfillment of
the requirements for the Research Pathway Project
Master of Information Technology

School of Computer Science
The University of Sydney
Australia

24 January 2023



THE UNIVERSITY OF
SYDNEY

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Xiaoxuan Li

Signature: Xiaoxuan Li **Date**: 2022/11/13

Abstract

This experiment investigates the effect that feature distillation has on the complex video interpolation model in order to make it more suitable for the requirements of industrial applications. The goal of the experiment is to make the video interpolation model more suitable for the requirements of industrial applications. We were successful in achieving real-time frame interpolation efficiency as well as a beneficial effect of model compression. The conclusion that feature distillation can increase the performance of video frame interpolation in terms of inference time and memory space has been verified by obtaining a distillation model.

In the ensuing trials including ablation, we came to the conclusion that the lessons to be learned from feature distillation are as follows: 1. setting distillation points closer to the end layer produces superior results to setting them closer to the beginning layer; 2. the number of the loss that was generated from the instructor models ought to be given more weight than the loss that was generated from the ground truth.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

Video permeates every element of human culture. The expansion of the pan-entertainment sector has prompted modern businesses to place greater emphasis on video quality, and there is a growing body of study in the subject of video enhancement.

Video enhancement is the process of improving the naked-eye appearance of a video by raising its resolution, lowering its noise, and increasing its frame rate. Video interpolation is a typical method for enhancing video. By constructing intermediate frames between two consecutive frames of a video, it boosts frame rate and improves visual quality. Video frame interpolation also enables a variety of applications, including slow-motion production, video compression, and the generation of training data for video motion deblurring. In addition, video frame interpolation algorithms operating at real-time speed on high-resolution video have further potential uses, such as higher frame rate playback on client players and video editing services for users with restricted computing capabilities.

In the field of video frame interpolation, a common model structure comprises an optical flow estimate network, a synthetic network, and a refinement network. The optical flow between two frames is estimated using an optical flow network. A synthetic network is used to build an intermediate frame, followed by a refinement network to enhance the intermediate frame's finer details. This conventional model structure has a number of significant flaws, including a reliance on the precision of optical flow estimation and a high degree of structural complexity.

This type of sophisticated model structure is intended to extract as much information as possible from the dataset during the training phase, which has historically been a common goal of machine learning research. However, industrial applications are more concerned with model size and inference time, and it is challenging to strike a balance between model precision, model size, and inference speed.

Knowledge distillation is presented as a solution for this type of issue. In the training phase, the model can be as broad and complex as necessary to encompass as much data as possible. In the application phase, a smaller model is provided to satisfy the model size and inference speed requirements. Knowledge distillation enables the lightweight model to learn from the bigger, pre-trained model, so enhancing the performance of the lightweight model and achieving a balance between model size, inference speed, and model accuracy.

CHAPTER 2

Literature Review

2.1 Video Frame Interpolation

2.1.1 Flow Estimation

Optical flow is the information about the motion of an object between consecutive frames that is computed using the change of pixels in the image sequence in the time domain and the correlation between adjacent frames to determine the correspondence between the previous frame and the current frame (Horn and Schunck, 1981).

Flow-Net (Dosovitskiy et al., 2015) is the first model developed to extract information about optical flow using neural networks. The input of Flow-Net (Dosovitskiy et al., 2015) consists of two consecutive images, the t frame and the $t + 1$ frame, which are first passed through systolic convolutional layers to extract their respective feature maps. For optical flow prediction, an expansion layer is used to enlarge the image to its original size after this step. Flow-Net's (Dosovitskiy et al., 2015) forecast accuracy does not surpass that of older optical flow estimation methods, but it has considerable advantages in terms of speed and stability. Flow-Net (Dosovitskiy et al., 2015) is an order of magnitude faster than standard approaches in terms of computing time.

PWC-Net (Sun et al., 2018) is influenced by the coarse-to-fine paradigm and employs a pyramidal structure to extract picture information. The network is built on three simple but time-tested principles: pyramidal processing, learning the fine optical flow of the current layer based on the previously acquired optical flow offset and developing a cost volume function. Through these three ideas, PWC-Net (Sun et al., 2018) surpasses Flow-performance Net's and achieves real-time.

Both Flow-Net (Dosovitskiy et al., 2015) and PWC-Net (Sun et al., 2018) are bound by the conventional Unet paradigm, which consists of a pair of encoders and decoders that are symmetric. RAFT (Teed

and Deng, 2020) deviates significantly from this Unet topology. Overall, the network consists of three components: Feature Encoder, Context Encoder, and Update Layer.

Two encoding layers, Feature Encoder and Context Encoder, adopt the same structure as the encoding layers of Unet. Multiple convolutional layers are employed to decrease the original map to one-eighth of its original size, hence decreasing the computation of the succeeding network. The Feature Encoder requires two input frames, both before and after the target frame, but the Context Encoder requires simply the one preceding the target frame.

The concept is straightforward: the Feature Encoder is used to extract the features of the optical flow, so the features are extracted from the before and after frames for the subsequent optical flow estimation, while the Context Encoder extracts contextual information, as the name suggests, ensuring that the estimated optical flow map maintains the same contextual information and position correspondence as the original one, so only the first frame is necessary.

In the Update Layer, Teed and Deng (2020) combine the output of the two encoders described above and iterate them simultaneously. RAFT (Teed and Deng, 2020) initially extracts the similarity between the findings encoded by the Feature Encoder for the two photos during the iteration. The similarity extends beyond the similarity of pixel values to include the similarity of feature descriptions. As a metric, RAFT (Teed and Deng, 2020) employs the simplest dot product similarity. A similarity pyramid is utilized to concentrate on the similarity at various scales, ensuring that both minute and large motions are seen simultaneously. RAFT (Teed and Deng, 2020) proposes the operation of table look-up to reduce the computational work required for similarity comparison, and it achieves greater accuracy in optical flow estimate than Flow-Net (Dosovitskiy et al., 2015) and PWC-Net (Sun et al., 2018).

2.1.2 Flow-based Models

Flow-based models are an important classification of video interpolation models. These models are based on the results of optical flow prediction for subsequent image generation, as their name suggests. Some of the representative ones are DAIN (Bao et al., 2019), RIFE (Huang et al., 2020), SuperSlomo (Jiang et al., 2018), Softmax Splatting (Niklaus and Liu, 2020), BMBC (Park et al., 2020), ABME (Park et al., 2021), and task-oriented flow (Xue et al., 2019).

DAIN (Bao et al., 2019) has the most conventional structure of all video interpolation models. DAIN (Bao et al., 2019) estimates optical flow and depth maps using PWC-Net (Sun et al., 2018) and MegaDepth

(Li and Snavely, 2018), respectively. Since optical flow and depth values are not monitored during the training of DAIN (Bao et al., 2019), DAIN (Bao et al., 2019) is simply initialized using the network weights that have been previously trained.

An essential premise in optical flow networks is that all motions are linear by default. DAIN's (Bao et al., 2019) answer is to apply the outside-in strategy, which fills the gaps with pixels from the surrounding area. DAIN (Bao et al., 2019) employs the image's depth information as supplemental data to estimate the optical flow. The depth information improves the efficiency of optical flow estimation since the projected flow prefers to sample pixels from closer objects and reduce the contribution of occluded pixels, whose depth values are greater. After obtaining the optical flow information, DAIN (Bao et al., 2019) uses a warping layer to generate intermediate frames and the following frame synthesis network to enhance the details.

DAIN (Bao et al., 2019) largely depends on the accuracy of optical flow estimation and picture depth information. Motion prediction distortion occurs in situations with intense action, and inaccurate depth information can generate blur and artifacts in some occlusions.

Unlike DAIN (Bao et al., 2019), RIFE (Huang et al., 2020) did not use a pre-trained optical flow network, but designed a new network IFNet to predict the optical flow. There are two main approaches in traditional optical flow networks, forward warping, and backward warping. Forward warping generates collision or holes, which requires additional operations to resolve the issues, for example, introducing depth information in DAIN (Bao et al., 2019). Backward warping requires a bidirectional flow of $Ft \rightarrow 0$ and $Ft \rightarrow 1$, which is computationally complex.

To solve both problems, RIFE (Huang et al., 2020) contains a new optical flow estimation network IFNet. IFNet consists of three parts: the $1/2$ times bi-linear resize layer, IFBlocks, the 2 times bi-linear resize layer. IFBlocks consist of 1/K times bi-linear resize layer, a convolutional layer, six residual blocks, a transpose convenlutional layer, and a K times bi-linear resize layer. The value of K changes for each layer.

It can be seen that IFNet uses a Coarse-to-Fine approach to gradually increase the resolution, similar to the architecture of PWC-Net (Sun et al., 2018). First calculating the coarse optical flow at low resolution makes it easier to capture large motions, and then calculating the fine optical flow at high resolution. The main component for optical estimation IFNet is IFBlock, and it also follows the pattern of the down-sampling and up-sampling modules.

In order to make the intermediate flow results predicted by IFNet more accurate, a pre-trained optical flow model is used to provide additional intermediate flow information as supervised information for training.

The privileged distillation proposed by RIFE (Huang et al., 2020) belongs to the category of knowledge distillation. Knowledge distillation is used to transfer knowledge from a large model to a small model. In privileged distillation, the teacher and student have the same architecture with different inputs. The teacher model gets more inputs than the student model, such as scene depth, and pictures from other camera views. With more inputs, the teacher model gets access to more information in the data and gets a more accurate representation to pass to the student model.

The fusion network of RIFE (Huang et al., 2020) consists of ContextNet, a context extractor, and FusionNet, which has an encoder-decoder architecture similar to U-Net. The context extractor and encoder parts of FusionNet have a similar architecture and consist of four ResNet blocks with a step size of 2. The decoder part of FusionNet has four transposed convolutional layers, and a sigmoid function is used to limit the output of FusionNet.

The optical flow prediction part of SuperSlomo (Jiang et al., 2018) uses backward warping. The idea of SuperSlomo (Jiang et al., 2018) is relatively simple, i.e., based on the assumption of uniform linear motion, and the following estimation method holds:

$$F_{t\to 0} = -(1-t)tF_{0\to 1} + t^2 F_{1\to 0}$$

$$F_{t\to 1} = (1-t)^2 F_{0\to 1} - t(1-t)F_{1\to 0}$$

This formulation estimates the bidirectional optical flows $F_{t\to 0}$ and $F_{t\to 1}$ by estimating the bidirectional optical flows $F_{1\to 0}$ and $F_{0\to 1}$ for the two input frames and combining them with the time t.

The computation here is done by an Unet. However, it is clear that this approach does not work well at the boundaries of the motion, because the motion is generally not uniformly linear at the boundaries of a moving object. The optical flow computed based on such assumptions can blur the edges of the object or cause artifacts.

To solve this problem, SuperSlomo (Jiang et al., 2018) uses a second Unet to perform the refinement of the optical flow. The inputs of the second Unet are six terms: two frames I_1 and I_0, optical flow $F_{t\to 1}$ and $F_{t\to 0}$ calculated by the first Unet, frame I_t calculated by I_1 and $F_{t\to 1}$, frame I_t calculated by I_0 and

$F_{t\to 0}$. and the output is the corrected value of $F_{t\to 1}$, $F_{t\to 0}$ and $V_{t\to 0}$, where $V_{t\to 0}$ is the visibility map of t for 0. Here it is required that the sum of $V_{t\to 0}$ and $V_{t\to 1}$ should be one. The corrected values of $F_{t\to 1}$ and $F_{t\to 0}$ plus the previously calculated $F_{t\to 1}$ and $F_{t\to 0}$ are the final $F_{t\to 1}$ and $F_{t\to 0}$. The edges of the network refined by the second Unet are sharper and the artifacts are relatively reduced, improving the accuracy of the optical flow estimation.

Another important feature of SuperSlomo (Jiang et al., 2018) is the implementation of arbitrary momentary frame interpolation. The previous methods can only achieve the estimation of intermediate frames at $t = 1/2$. If want to insert more frames in between two frames, it is required to perform prediction multiple times. For example, to estimate the intermediate frames at $t = 1/4$, we need to estimate the intermediate frames at $t = 1/2$ first. It can be seen that the computation is serial because some intermediate frames cannot be computed until the other frames are completed, so it will be more time-consuming. This method also has no way to estimate the intermediate frames at $t = 1/3$. In addition, the errors in the optical flow estimation are accumulated in the recursive operation.

To achieve arbitrary momentary interpolation, SuperSlomo (Jiang et al., 2018) introduces a time variable t to control the generation process of intermediate frames. Neither the parameters of Super Slomo's (Jiang et al., 2018) optical flow computation network nor the interpolation network depends on the specific time step of the interpolated frame, and the method can generate as many intermediate frames as desired in parallel.

For SuperSlomo (Jiang et al., 2018), one of the most important problems is to find a suitable dataset. Previous datasets used for training video interpolation frames tend to be around 30fps, too low a frame rate can actually result in motion that is not uniform between frames. It is sufficient for training a model that only estimates 1/2 most of the time. However, for interpolation in an arbitrary given time, it is necessary to ensure that the frame rate of the video is high enough to be used as the training dataset.

To train SuperSlomo (Jiang et al., 2018), the authors collected multiple 240 fps videos from YouTube and handheld video cameras. In total, 1100 video segments were collected, consisting of 300,000 individual video frames at 1080×720 resolution. The dataset used in the paper is the 240fps one, so it can be approximated as a uniform action.

The goal of the optical flow algorithm is to make the warped image consistent with the target image. However, this exact optical flow estimation is premised on the assumption of consistent luminance, which is not always accurate in challenging situations such as changing illumination and pose, resulting

in blurred target boundaries. In addition, this image optical flow estimation consistent with object motion changes is not applicable to all video processing tasks, and small errors in the optical flow field can lead to artifacts in the interpolated frame results.

The TOFlow (Xue et al., 2019) therefore improves the part on optical flow estimation, with the main difference that instead of training optical flow separately, the pre-trained optical flow module is trained jointly with subsequent processing to learn the optical flow best suited to express the features for a given task. The model uses an optical flow-based approach to implement three tasks: video frame interpolation, video denoising, and video super-resolution, with the advantages of good video processing, low computational effort, and the ability to be self-supervised.

The TOFlow (Xue et al., 2019) model structure consists of three parts: Flow Estimation, Transformation, and Image Processing. In the video interpolation task, the number of input frames is N=3; in the denoising and super-resolution tasks, the number of input frames is N=7. The Flow Estimation module takes a pre-trained SPyNet (Ranjan and Black, 2017) as the backbone, with N-1 SPyNet (Ranjan and Black, 2017) optical flow networks with the same structure and shared parameters.

It is worth noting that the official SPyNet model does not contain a batch normalization structure, while the SPyNet (Ranjan and Black, 2017) model used in the TOFlow (Xue et al., 2019) paper adds a batch normalization structure with a batch size of 1. In the video interpolation task, the reference frame is the frame to be generated and therefore is not included in the input. In the interpolation task, the TOFlow (Xue et al., 2019) network uses SPyNet (Ranjan and Black, 2017) to process frames 1 and 3 to obtain If1→f3 and If3→f1. With the optical flow field predicted by the Flow Estimation module, the transformation module uses the flow warp function to register the input frame to the reference frame. In the video interpolation task, this part acquires the mapping of frame 1 and frame 3 to frame 2:

$$I_{f1\to f2} = I_{f1\to f3} \times 0.5$$

$$I_{f3\to f2} = I_{f3\to f1} \times 0.5$$

The Image Processing module uses the ResNet (He et al., 2016) structure to process the mapped images acquired by the Transformation module into the final interpolation result.

2.2 Knowledge Distillation

In actual applications, deep learning presents numerous obstacles. To handle complex learning problems, deep learning network models are frequently developed with a significant number of deep layers and parameters. The training and deployment of these models necessitate substantial processing resources, and it is difficult to directly apply them to the most common embedded and mobile devices.

Industrialization will move gradually to intelligence, and the emergence of edge computing indicates that AI will integrate progressively with miniaturized and intelligent devices, necessitating models that are more practical, efficient, and lightweight to accommodate the deployment of these devices. The field of video interpolation also faces similar difficulty, therefore model compression is a crucial component.

A typical knowledge distillation framework comprises three components that are trained on a supervised data set: the Teacher model, the student model, and the knowledge transfer. There are three types of knowledge distillation: logits(response)-based knowledge distillation, feature-based knowledge distillation, and relation-based information distillation.

Logits(response)-based knowledge acquires knowledge directly from the instructor model's output layer. Relation-based is intended to understand the relationship between input-hidden-output, whereas Feature-based acquires knowledge from hidden intermediate layers. Three classical methods with distinct types of knowledge distillation are presented individually in the following sections.

2.2.1 Logit Distillation

Knowledge distillation is widely used in model compression and migration learning. Cristian et al. (2006) first proposed the idea of compressing models by knowledge distillation, but no actual work was elaborated. Hinton et al. (2015) first formally defined distillation and proposed a corresponding training method. In the article *Distilling the Knowledge in a Neural Network*, the motivation of Hinton et al. (2015) is to find a way to distill the knowledge of multiple models to a single model. This is the first experiment in the field of logits(response)-based knowledge distillation. This paper defines the field.

It is generally believed that the parameters of a model retain the knowledge learned by the model, so the most common way of transfer learning is to do pre-training on a large dataset first, and then use the parameters obtained from pre-training to fine-tune them on a smaller dataset. The two datasets often

have different domains or different tasks. For example, pre-training is done on Imagenet (Krizhevsky et al., 2017) first and then detection is done on the COCO dataset (Lin et al., 2014).

Hinton et al. (2015) suggest that models can be understood as black boxes and knowledge as a mapping relationship between input and output. Therefore, we can first train a teacher network, then utilize the output result q of the teacher's network as the aim of the student network, and train the student network so that its output is close to q. Therefore, the loss function can be expressed as:

$$L = CE(y, p) + CE(q, p)$$

Here, CE represents the Cross-Entropy (CE), y represents the one-hot encoding of the real label, q represents the output of the teacher network, and p represents the output of the student network.

However, it may not be suitable to use the output result q of the teacher network's soft-max directly. After a network has been trained, there will be a high level of confidence in the correct answer. For instance, based on MNIST dataset (Deng, 2012), the prediction probability for some inputs of 2 will be high, yet the prediction probabilities for similar inputs, such as 3 and 7, are 10-6 and 10-9. Thus, it is difficult for the teacher network to learn similar information about the data to communicate with the student network. Due to the fact that their probabilities are close to 0, Hinton et al. (2015) therefore presented softmax-T with the following equation:

$$q_i = \frac{exp(z_i/T)}{\Sigma_j exp(z_j/T)}$$

Here, q_i is the object learned by the student network (soft targets), and z_i is the neural network's logit output before softmax. If T is assumed to be 1, this formula is softmax and logit is used to calculate the probability of each category. If T is close to zero, the greatest value will be closer to one and the remaining values will be close to zero, which approximates one-hot coding. If T is greater, the distribution of output results is smoother, which corresponds to a smoothing function and serves to retain similar information. If T equals infinity, the distribution is uniform.

The concluding piece trains the network using the aforementioned loss function. On the MNIST dataset (Deng, 2012), the large network is trained with 67 test set errors before the tiny network is trained with 146 test set errors. Adding soft targets to the objective function, which corresponds to the canonical term, decreased the number of test set mistakes to 74. This illustrates that the instructor network transfers knowledge to the student network, hence improving the outcomes.

The second experiment was conducted in the voice recognition domain, where 10 DNNs were trained using different settings, and the results of emsemble shown some improvement over the individual models. Then, these 10 models are utilised to train the student network as a teacher network. The resultant Distilled Single model exhibits some improvement over the direct single network.

Hinton et al. (2015) reached the conclusion that knowledge distillation can transmit information from one network to another, regardless of whether the two networks are homogeneous or heterogeneous. This is accomplished by first training a teacher network and then training the student network utilising the output of the teacher network and the true labels of the data. Information distillation can be used to transform a huge network into a small network with comparable performance, or to transfer the gained knowledge from numerous networks into a single network so that the performance of the single network is comparable to the results of ensemble.

Moving from powerful huge networks or integrated networks to small networks with simple structures and rapid operation is the classic concept for distillation. Zhang et al. (2018) breaks this predefined "strong-weak relationship" and introduces Deep Mutual Learning, which enables a group of student networks to learn from and guide each other during the training process, as opposed to a predefined static one-way transition path between professors and students. Zhang et al. (2018) add Kullback Leibler (KL) Divergence to quantify the output probabilities $p1$ and $p2$ of two student networks. The KL distance is a measure of how well two probability distributions match, and the higher the disparity between the distributions, the greater the KL distance. The authors utilize the KL distance to determine if the predictions of the two networks, $p1$ and $p2$, are consistent.

The KL distance from p1 to p2 is computed as:

$$D_{KL}(p_2 \parallel p_1) = \sum_{i=1}^{N} \sum_{m=1}^{M} p_2^m(x_i) log(p_2^m(x_i)/p_1^m(x_i))$$

The final trained loss function is the sum of the cross-entropy Loss and KL scatter Loss for each of the two student networks that were trained independently. According to the experimental findings, the effect is evident, the increase is steady, and the effect increases linearly with the number of student networks.

Cho and Hariharan (2019) experimentally observed that the teacher model with the better performance does not distill the student model with the better performance; they hypothesized that the capacity mismatch is the reason why the student model does not mimic the teacher model but instead brings out

the main loss, and they proposed an early-stop teacher regularisation for distillation. The regularisation of teachers is advocated, and the distillation should be terminated early when convergence is near. As demonstrated in Table 2.1, the student distillation effect is not necessarily enhanced as the instructor network grows.

Teacher	**Teacher Error(%)**	**Student Error(%)**
-	-	
ResNet18	30.24	30.57
ResNet34	26.70	30.79
ResNet50	23.85	30.95

TABLE 2.1: Top-1 error rate for various teachers for a ResNet18 student on ImageNet. Adapted from the original paper *On the efficacy of knowledge distillation.*

The specific early-stop teacher regularisation approach is not specified by Cho and Hariharan (2019), but it can be inferred that it is based on the training loss curve. If the training loss curve converges then it would be the end point of distillation.

Xie et al. (2020) continue to use the knowledge distillation softened labels, but focus on the data problem and train the student model using a larger noisy data set.

The motivation of Xie et al. (2020) is to use less labeled data to further utilize large-scale unlabeled data for semi-supervised or self-supervised learning. Simply put, the teacher model is used to generate pseudo-labels to train the student model, and the student model is made to outperform the teacher model by adding noise, and the process is iterated to obtain a more robust model.

The approach Xie et al. (2020) used is to first train the teacher network on ImageNet (Krizhevsky et al., 2017), then use the trained noise-free teacher network to generate as accurate pseudo-labels as possible for another dataset, JFT dataset (Sun et al., 2017), and finally use the pseudo-labeled dataset JFT dataset (Sun et al., 2017) and ImageNet (Krizhevsky et al., 2017) to train the student model and the new student is reused as a teacher model for next iteration. Xie et al. (2020) repeat the above steps three times in total.

Xie et al. (2020) gain many valuable insights during student network training. The first is that increasing the dataset noise can improve the robustness and generalization ability. The second trick is data filtering,

which filters out images with low confidence in the teacher model, as this usually represents out-of-domain images. Data balancing, which balances the number of images from different categories, is another technique useful for training. The use of soft labels for the output of the teacher model is also helpful for the experiments.

Xie et al. (2020) conclude that the student model works better if it is larger than the teacher model. If the size is the same, using only noisy student training can also have a significant improvement.

Yang et al. (2019) continue to employ knowledge distillation softened labels, but add limitations to the distillation process to reach the optimization objective. Restrictions are added to the teacher model or student model. Yang et al. (2019) discovered that, in addition to the ground truth class, the secondary class can effectively learn inter-class similarity and prevent the student network from over-fitting in order to get good outcomes.

A number of classes are picked by Yang et al. (2019) with the greatest confidence scores and assumed that these classes are more likely to share semantic similarities with the input image. A constant integer K was assigned to indicate the number of semantically plausible classes, inclusive of the ground truth class, for each image. The difference between the ground truth class and the other $K - 1$ classes with the highest scores was then determined. It is found out that the larger the K value is, the better the result (Yang et al., 2019).

Usually, knowledge distillation performs poorly when there is a large difference in the capacity of student and teacher network models. Phuong and Lampert (2019) adapt the muti-exit architectures to do ensemble distribution knowledge distillation. The ensemble distribution knowledge distillation method can well ensure the distribution diversity, meanwhile, the fusion of multiple model structures is better.

Logits(Response)-based knowledge distillation possesses both benefits and drawbacks. The benefits include being simple and easily understood, and implementation for the output layer is basic and straightforward. The student model learns the probability distribution of the output of the teacher model, which is comparable to supplying information about the similarity between categories, making it easier to learn and offering additional supervisory signals.

It also has the following disadvantages. For example, distillation efficiency is dependent upon the calculation of softmax loss and the number of classes. Moreover, there is no solution to the problem of lacking

data labels and it is difficult to successfully extract information from the instructor network when the student network model is too small.

2.2.2 Feature Distillation

FitNet (Adriana et al., 2015) is presented in the context of the work *Distilling the Knowledge in a Neural Network* of Hinton et al. (2015), and the main idea is to use a wide and deep teacher model to train a narrow and deep student model.

Previous knowledge distillation methods mainly train the teacher network to a shallower and wider network, without taking full advantage of the depth. And this article makes an attempt to learn knowledge through intermediate feature layers for the first time and use intermediate features for knowledge distillation, so this article is considered the originator of the feature distillation area.

Adriana et al. (2015) first choose the output of the N_{th} layer of the teacher model feature extractor as the hint layer, and the parameters from the first layer to the N_{th} layer correspond to W_{hint} in the graph, and then choose the output of the M_{th} layer of the student model feature extractor as guided, and the parameters from the first layer to the M_{th} layer correspond to W_{Guided} in the graph. Feature map dimensions of the student model and the teacher model may not match, so a convolutional layer adjuster, denoted as r, is introduced in the student network to adjust the dimensionality of the guided layer.

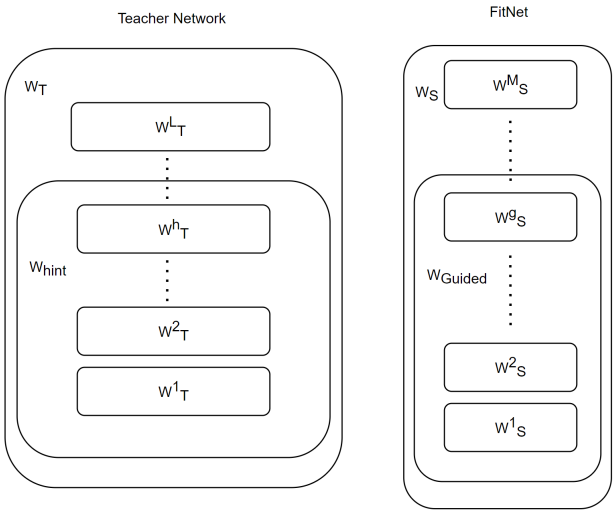


FIGURE 2.1: Distillation architecture of FitNet

In the training process of the student model, the feature loss function is first minimized. The equation of the feature loss function is:

$$L_{HT}(W_{Guided,W_r}) = 0.5\times \parallel u_h(x;W_{Hint}) - r(v_g(x;W_{Guided};W_r)) \parallel^2$$

Where u_h denotes the function corresponding to the teacher model from the first layer to the Nth layer, v_g denotes the function corresponding to the student model from the first layer to the M_{th} layer, and r denotes the convolutional layer adjuster, and the corresponding parameter is noted as W_r. Because there is no label information in this stage, the distillation granularity is not fine enough, so the paper introduces the second stage of training, which is the distillation of the student model using the knowledge distillation proposed by Hinton et al. (2015). The experiments achieved great results and started research in the field of feature distillation.

Zagoruyko and Komodakis (2016) applied image attention in feature distillation. In this paper, the student model is led by extracting the attention graph generated by the teacher model, so that the student model's attention graph resembles that of the teacher model. This allows the simple model to not only learn feature data but also comprehend how to refine feature data. It makes the student model's generated features more versatile and not restricted to the teacher model.

2.2.3 Relation Distillation

Instead of fitting the output of the Teacher model, the interactions between the Teacher model's layers are de-fitted, analogous to a teacher instructing a student how to solve a problem, where the intermediate outputs are unimportant and the process lead to solution should be taught in more detail. This is the methodology of Yim et al. (2017).

The relationship between layers is specified by the inner product. If layer A has M output channels and layer B has N output channels, then an $M \times N$ matrix is formed to describe the relationship between the two layers, where each element (i, j) is the inner product of the i_{th} channel in layer A and the j_{th} channel in layer B. Therefore, this method requires the same shape of the feature map in both layers. The authors use the residual module to avoid the identical calculation of spatial size.

In the article, this matrix is referred to as an FSP matrix (flow of solution procedure matrix), and it is also a $Gram$ matrix. Each value of the Gram matrix can be interpreted as representing the degree of correlation between the feature maps of channels i and j. x represents the input picture, W represents

the FSP weights, $F1/F2$ is the resultant feature map, hw represents the height and width of the feature map. m/n represents the channel. The computation equation is

$$G_{i,j}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w}\frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,i}(x;W)}{h \times w}$$

Specific steps Yim et al. (2017) take are standard. First minimize the L2 Loss between the FSP matrix of the teacher model and the FSP matrix of the student model, which is used to initialize the trainable parameters of the student model. Then fine-tune the student model on the dataset of the target task. The loss function is:

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n}\lambda_i \times \parallel G_i^T(x;W_t) - G_i^S(x;W_s) \parallel_2^2$$

Where λ represents the weight coefficients of different layers/points, which are set to the same weights by Yim et al. (2017).

If two inputs have highly similar activations in the teacher network, it would be advantageous to direct the student network towards a combination of parameters that also produce highly similar activations for that input, for the students to better learn the capabilities and knowledge of the teacher network, and vice versa.

Based on this observation and hypothesis, Tung and Mori (2019) proposed a preserved similarity loss to induce the student network to learn the knowledge of the teacher network in relation to the relational representation within the data.

Tung and Mori (2019) conduct experiments to indicate the results obtained by plotting the vector obtained by computing the mean value within all channels in the last convolutional layer of the teacher network for each of the 10,000 images in CIFAR-10. By dividing into ten categories, each corresponding to 1000 adjacent images, it is revealed that the activations of the 1000 adjacent images are similar, while there are significant differences between the different categories.

From the above study, it can be seen that relation-based knowledge distillation generalizes better, and the current state-of-art methods are based on feature-based knowledge distillation or relation-based knowledge. Also, relation-based knowledge distillation can handle cross-domain transfer and low-level vision problems.

There are still some problems with relation-based knowledge distillation, such as it is difficult to measure information loss, so it is difficult to choose the best method. Most of the methods choose intermediate layers randomly, which is not interpretable. Also, the distillation position of features needs to be selected manually based on the task.

CHAPTER 3

Evaluation

3.1 Datasets

There are three datasets used in this experiment, Vimeo90K (Xue et al., 2019), UCF101 (Soomro et al., 2012), SNU-FILM (Choi et al., 2020).

Vimeo90k (Xue et al., 2019): This experiment uses the test set of Vimeo90K (Xue et al., 2019) dataset, which has 3782 triples with 256×448 resolution for each image.

UCF101 (Soomro et al., 2012): The part selected for this experiment is the test set of UCF101 (Soomro et al., 2012) on video frame interpolation model, which has been introduced to video frame interpolation field by DVF (Liu et al., 2017). This dataset consists of 379 triplets, and the resolution of each frame is 256×256.

SNU-FILM (Choi et al., 2020): This dataset was proposed by CAIN (Choi et al., 2020). The original data consists of eleven videos from the GOPRO test set (Nah et al., 2017) and twenty YouTube videos. There are four levels of difficulty: easy, medium, hard, and extreme, depending on the magnitude of the movement. The test set contains a total of 1240 triplets, with a resolution of 1280×720 per frame.

3.2 Metrics

Two common metrics, Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM), are used to evaluate the quality of the generated frames. They are widely used in the field of video interpolation.

PSNR is one of the most common and widely used objective image evaluation metrics. It is based on the difference between the corresponding pixel points. However, human eyes are more sensitive to contrast differences in luminance than chromaticity and are more sensitive to contrast differences at lower spatial

frequencies. Moreover, human eyes' perception of a region is affected by its surrounding neighboring regions. Since PSNR does not take into account the visual characteristics of human eyes, the evaluation results are often inconsistent with human subjective perception.

PSNR is defined by the equation:

$$PSNR = 10 \times log_{10}(\frac{MAX_I^2}{MSE})$$

Where MAX_I is the max number of image color and MSE is the mean squared error between two images K and I size $m \times n$.

SSIM measures image similarity in terms of brightness, contrast, and structure, which are in alignment with human eye observation. It is a compliment of PSNR.

SSIM is defined by the equation:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where μ_x is the average of x and similarly μ_y is the average of y. σ_{xy} is the covariance of x and y, σ_x^2 is the variance of x and σ_y^2 is the variance of y.

Since model compression result and model efficiency are also important in this experiment, we take model size, the number of parameters and inference time into consideration as well.

3.3 Implementation Details

For this experiment, the training split of Vimeo90k (Xue et al., 2019) is chosen to train the student model. Pytorch is used for implementation and the model is trained on a single Tesla V100 graphic card. Other training settings are maintained in accordance with the initial Channel Attention for frame Interpolation model (Choi et al., 2020). Adam (Kingma and Ba, 2014) is selected as the optimizer.

The teacher model utilized in the studies is an official pre-trained model made available to the public by Channel Attention for frame Interpolation model original paper (Choi et al., 2020).

The model is trained for 200 epochs over a period of approximately two days. Random vertical and horizontal flipping are utilized for data argumentation. Random temporal order swappings are done to

two input frames. The initial learning rate is 10^{-4}, which is reduced by a factor of 2 when the validation loss reduction reaches a plateau.

CHAPTER 4

Experiments

4.1 Baseline Model - Channel Attention for frame INterpolation (CAIN)

In this particular experiment, Channel Attention for frame Interpolation (CAIN) (Choi et al., 2020) is selected to serve as the primary support backbone. In the field of video frame insertion, CAIN (Choi et al., 2020) is considered to be one of the most important models. It is capable of producing state-of-the-art outcomes by addressing channel attention.

CAIN (Choi et al., 2020) accepts the first and third frames as input and generates an intermediate frame as output. The structure of CAIN (Choi et al., 2020) is intuitive and straightforward, consisting of pixel shuffle and residual groups with channel attention. Its network structure is depicted in Figure 4.4.

Pixel shuffle is used to replace the conventional encoder-decoder structure, as it is able to preserve a large number of the original image's details with minimal effort. It has the advantage of expanding the receptive field without sacrificing local information. This component is symmetric, similar to the traditional encoder decoder U-net, to perform a down shuffle at the beginning of the model and an up shuffle after passing through the entire network and receiving intermediate frames. The absence of learnable parameters in pixel shuffle reduces the computational cost and training difficulty of the model.

The residual groups with channel attention refer to a collection of convolutional neural network structures comprising ResNet (He et al., 2016) as the backbone. This is the central component of CAIN (Choi et al., 2020), through which image features are extracted and analyzed.

Figure 4.1 depicts the specific inside structure of one residual group with channel attention. In order, each residual channel attention block is composed of a 3×3 convolution layer, a rectified linear unit activation layer, another 3×3 convolution layer, and a channel attention module. The channel attention module is comprised of one global average pooling layer, which aggregates channel-specific

variables, and two fully-connected layers. The final output of the sigmoid function F_{attr} is calculated as an element-by-element product of the input feature F.

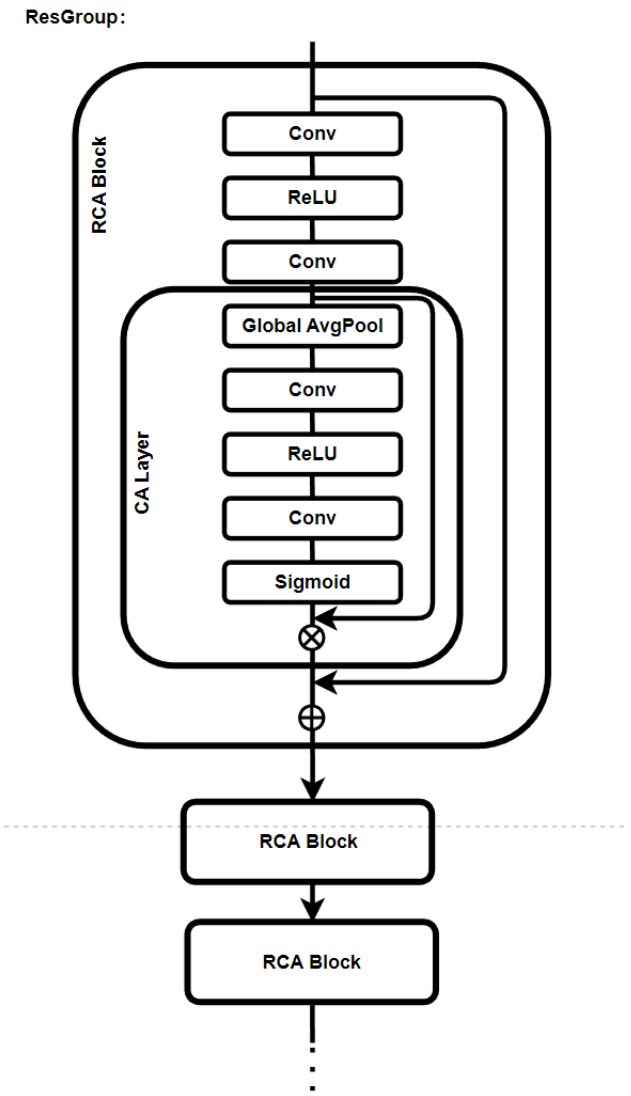


FIGURE 4.1: Network architecture of one residual group

Conventional video interpolation typically consists of a number of components as depicted in Figure 4.2. Specifically, a network for optical flow estimation, an approximation refinement network, and a

synthesis network are the most representative network structure stack. This kind of model structure is unsuitable for experimental studies since the effect of improving one specific component of the model may influence other components, which makes it difficult to directly reflect the improvement in the outcome.

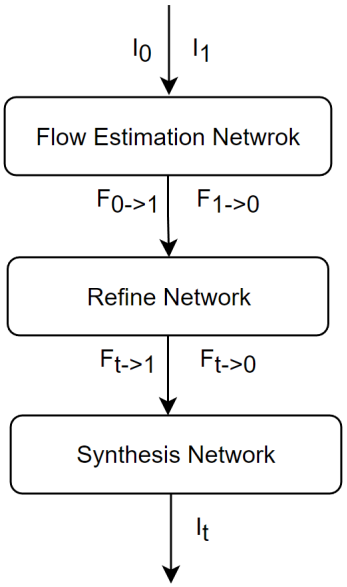


FIGURE 4.2: Typical Architecture of existing video frame interpolation models

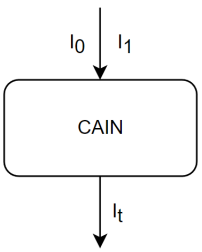


FIGURE 4.3: Architecture of CAIN

CAIN (Choi et al., 2020) has a relatively simple structure compared to other video interpolation models, with only two major components: pixel shuffle and groups of residual channel attention blocks.

This feature makes this experiment more convenient, as variables can be easily controlled. In addition, CAIN's (Choi et al., 2020) primary component contains recurring patterns. This is a good starting place for investigating the model's redundancy.

CAIN (Choi et al., 2020) also is an end-to-end model that enables direct inferences to be drawn by observing the quality of the output image and is ideally suited for experimental investigations.

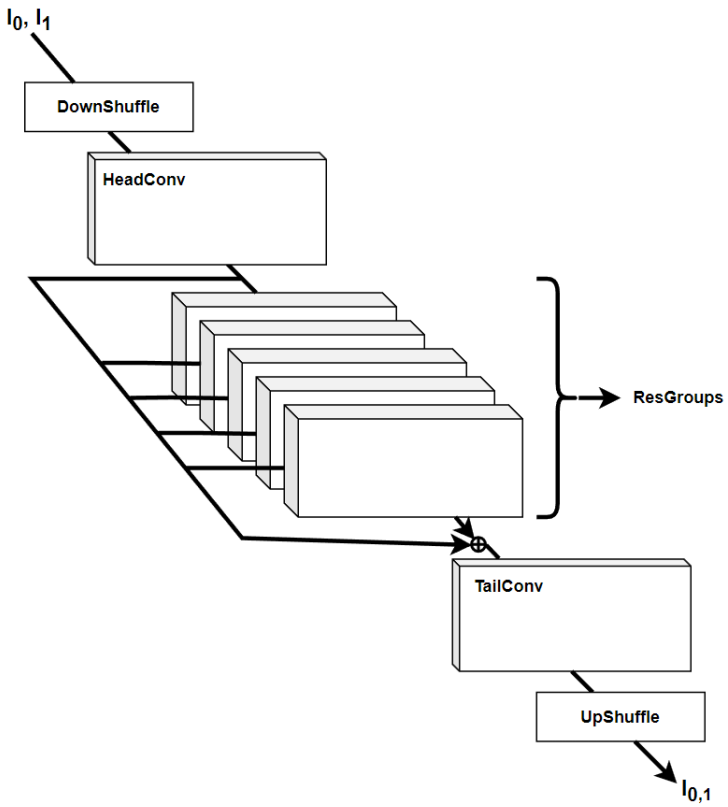


FIGURE 4.4: CAIN model architecture

4.2 Baseline Model Adjustment

Due to the back-propagation property of neural networks, the gradient updates are faster for the layers near the output and slower for the layers far from the output. If the network has more layers, the gradient may disappear or explode in the deeper network due to the accumulation of gradient updates layer by layer.

In this experiment, the main part of CAIN (Choi et al., 2020) has sixty-two layers, which is a deep neural network, and there is a certain risk of gradient disappearance or gradient explosion. At the same time, CAIN (Choi et al., 2020) uses sigmoid as the activation function in the channel attention layer.

The problem with the sigmoid function is that it converges to zero or one for large positive or large negative inputs, and the derivative is very close to zero. Therefore, when the back-propagation algorithm intervenes, it has virtually no gradient to back-propagate through the network. And as the algorithm moves down through the top layer, any small amount of residual gradient that exists keeps diluting, eventually causing the gradient to disappear.

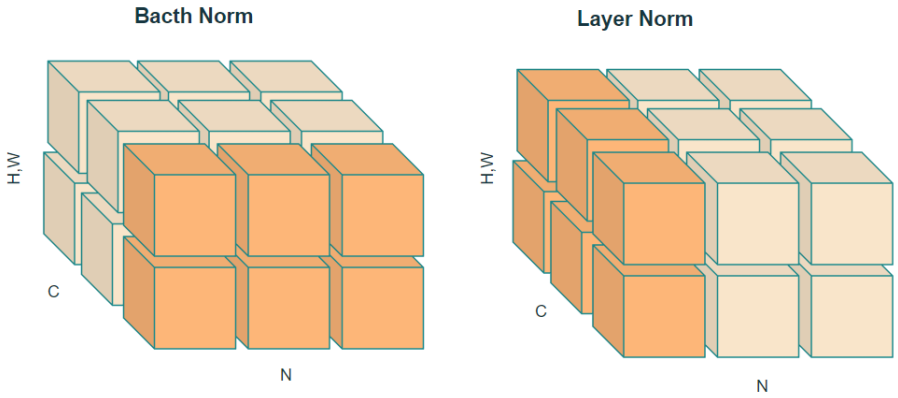


FIGURE 4.5: Batch Normalization and Layer Normalization

A common solution for gradient explosion or disappearance is batch normalization. Batch normalization has the effect of accelerating the convergence speed of the network and improving the training stability. It ensures the stability of the network by normalizing the output distribution to have a mean of zero and a variance of one. Specifically, batch normalization reduces the internal co-variate shift by normalizing the inputs of each layer, fixing the mean and variance of the input signals of each layer, and preventing

gradient diffusion, so that the output distribution of the model and the output distribution of the data in the target space is consistent as much as possible.

In this experiment, we consider that the input images are the first and third frames of triplets, and the first frames or the third frames of each triplet are disjoint, so if they are normalized together, they could interfere with each other and degrade the model performance. Therefore, instead of layer normalization, we choose layer normalization. Because the channels of the images belong to similar classes, layer normalization does not degrade the CAIN model's (Choi et al., 2020) expressiveness, and also reduces the risk of gradient explosion or gradient disappearance.

Layer normalization plays a similar role to batch normalization. The difference between the two is that the scope of batch normalization is a mini-batch of input samples, while the scope of layer normalization is all dimensions of each input sample, as shown in Figure 4.5.

Another interesting fact of this experiment is the version of Pytorch version could influence the model training process. With Pythorch 1.3, the training process stays in a more stable position and the chance of gradient vanishing is lower.

4.3 Loss Function

The generic design of feature distillation is in form:

$$L_{distill} = d(T_t(F_t), T_s(F_s))$$

Where T_t refers to the teacher feature transformation function. T_s refers to the student feature transformation function. d represents the distillation function, and the resultant output finally obtains the distillation loss $L_{distill}$. The teacher feature transformation function, student transformation function, and the distillation function together lead to the distillation loss.

The teacher features transformation function acts as a preprocessing step to reveal the information in the teacher model. It makes the information of the teacher model more easily received by the student model, thus improving the result of feature distillation. The student features transformation function usually makes the number of channels of the student model rise, to achieve the main purpose of matching the number of channels of the teacher model, as shown in Figure **??**.

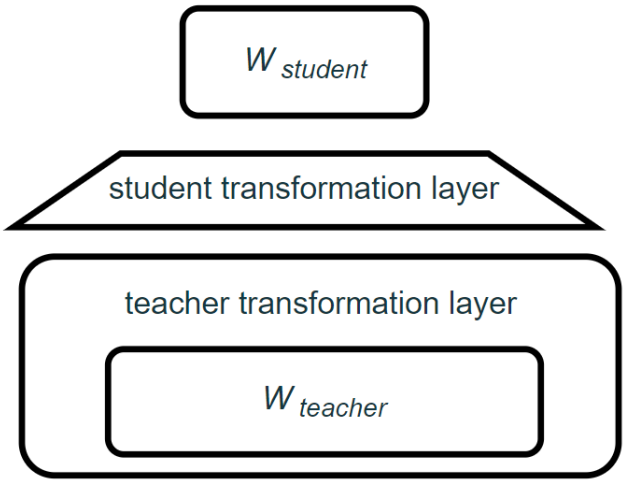


FIGURE 4.6: General form of teacher transformation and student transformation

The common teacher features transformation function, such as the attention mechanism of AT (Zagoruyko and Komodakis, 2016), and the gradient processing of Jacobian (Srinivas and Fleuret, 2018), both reduce the teacher model feature vector dimension. This leads to the loss of information in the teacher model, which affects the learning of the student model.

In this experiment, the pre-trained model of CAIN (Choi et al., 2020) is used as the teacher model and the reduced layer version of CAIN (Choi et al., 2020) is used as the student model. Since the neural network layers of the teacher model and the student model are identical, there is no need to trim the teacher network, and the alignment with the number of channels of the student network can be achieved directly. Reducing the teacher features transformation can also significantly reduce the information loss during knowledge distillation to achieve the best distillation effect. Therefore, in this experiment, the teacher features transformation function is not used. Correspondingly, the student features transformation is also removed together with the network parameters due to the reciprocity.

The distillation function often used in classical knowledge distillation methods is the L2 distance and its variants. For example, FitNet (Adriana et al., 2015), AT(Zagoruyko and Komodakis, 2016), Jacobian (Srinivas and Fleuret, 2018) all use L2 distance. From the results of these experiments, L2 distance can better reflect the difference between the teacher model and the student model, so in this experiment, the distillation function in this experiment follows the L2 distance, which is written as $d = \| F_t - F_s \|^2$.

The loss function for training is defined by:

$$L_{total} = \alpha L_{GroundTruth} + \beta L_{distill}$$

Where

$$L_{GroundTruth} = \| I_{student} - I_{GroundTruth} \|_2$$

$$L_{distill} = \| I_{student} - I_{teacher} \|_2$$

and $\alpha = 1 \times e^{-5}$, $\beta = 1 \times e^{-4}$.

4.4 Distillation Details

4.4.1 Number of Distillation Points

The number of distillation points affects the learning effect of the student model. In this experiment, distillation points set at the last layer of each set of residual networks, the head convolution layer, and the tail convolution layer worked best. Contrary to intuition, the number of distillation points is not as high as it could be. Setting extra distillation points in the middle of each set of residual networks will make the results worse. Specific experiments are discussed in the ablation experiments of this paper.

4.4.2 Position of Distillation Points

Distillation points can be placed at any layer in the network theoretically. Intuitively, the more distillation positions are placed towards the back side of the network, the more knowledge can be learned. experimental results of FitNet (Adriana et al., 2015) proved that distillation from any intermediate point gives poor results, so instead of setting the distillation position in the middle layers as in FitNet (Adriana et al., 2015), the main distillation point is set at the last layer of each residual group in this experiment.

The specific position is shown in Figure 4.7. It is proved by ablation experiments that this approach can indeed achieve better results.

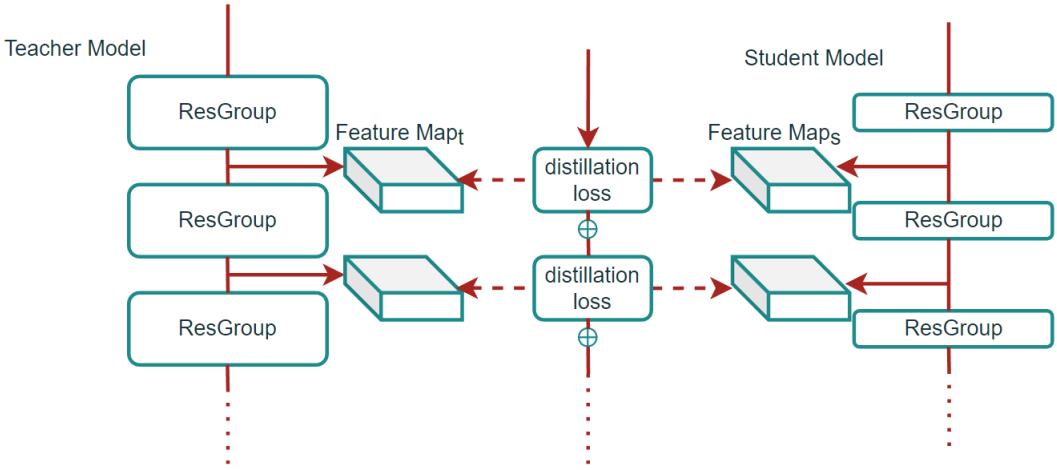


FIGURE 4.7: Feature distillation position illustration

CHAPTER 5

Results

5.1 Main Results

5.1.1 Quantitative Evaluation

Due to the fact that this experiment investigates the efficacy of knowledge distillation in the field of video frame insertion, the results section examines various aspects, such as the student model's compression efficiency, precision, model size, and inference time.

By comparing the original pre-trained model (Choi et al., 2020), the student model with knowledge distillation after layer cut, and the model with direct layer cut and no distillation, Table 5.1 illustrates the benefits of the feature distillation strategy.

The experiment is run on a test graphic card Tesla V100, and the input data set is UCF101 (Soomro et al., 2012). The input image resolution is adjusted to 256×256. According to the table, the PSNR and SSIM differences between the distilled student model and the original model are only 0.16 and 0.0007, showing that the student model is nearly as accurate as the original model.

In terms of model size, the number of parameters in the distilled student model is less than one-third of the original model. In addition, the average inference time per frame of the student distillation model on the UCF101 dataset (Soomro et al., 2012) is significantly less than that of the original teacher model (Choi et al., 2020), which requires 0.669ms for every generated frame, but the student model only requires 0.103ms per frame.

This experiment demonstrates that feature distillation can greatly compress the model while preserving the original model's correctness to the greatest extent. In contrast, despite the fact that the model with direct layer cut but no distillation has a similar size and speed as the model with distillation, it can only obtain PSNR 18.75 and SSIM 0.5313, which barely meets the requirements.

	original	**distilled**	**without distillation**
PSNR	34.95	34.79	18.75
SSIM	0.9679	0.9672	0.5312
parameter numbers(M)	42.8	12.6	12.6
Time (ms)	0.669	0.103	0.213

TABLE 5.1: Feature distillation compression results

Table 5.2 presents a comparison between the student model and CDFI (Ding et al., 2021), FLAVR (Kalluri et al., 2020), and RRIN (Li et al., 2020). The experiment is conducted on a testing graphic card Tesla V100, and the input dataset is UCF101 (Soomro et al., 2012). The image resolution is 256×256.

In terms of accuracy, the distilled student model is comparable to the state-of-art models. It exceeds RRIN (Li et al., 2020) and is slightly inferior to CDFI (Ding et al., 2021) and FLAVR (Kalluri et al., 2020).

The distilled student model outperforms other models in terms of runtime memory consumption. It consumes the least amount of memory space 1473 MB, 2165 MB less than CDFI (Ding et al., 2021), and 1124 MB less than FLAVR (Kalluri et al., 2020).

It is worth noting that our model is the fastest in terms of inference time per frame, only 0.103ms. As a point of comparison, FLAVR (Kalluri et al., 2020) is the slowest model, with an average inference time of 1.538ms per frame.

model	**inference time(ms)**	**memory(MB)**	**PSNR**	**SSIM**
CDFI	0.2931	3638	35.21	0.9673
FLAVR	1.538	2594	34.97	0.9680
RRIN	0.188	2656	32.67	0.9666
CAIN (original)	0.669	2065	34.95	0.9679
CAIN (distilled)	0.103	1473	34.79	0.9672

TABLE 5.2: Distilled CAIN compares with state-of-art models on UCF101 dataset

Fig.3 depicts a comparison between the simplified student model and state-of-the-art models on various datasets using the Tesla V100 test graphics card. Evidently, DAIN and BMBC are more precise and perform well on the majority of datasets. Their disadvantage is the lengthy inference time. On the 1280x720 dataset, DAIN and BMBC have average inference times of 1.033s and 3.845s, respectively. Our model's accuracy is comparable to that of the state-of-the-art, but it runs far faster, with an average inference time of only 0.014s on the 1280x720 resolution test input data.

Model	SNU-FILM				Vimeo90k	UCF101
	Easy	Medium	Hard	Extreme		
DAIN	39.73/0.9902	35.46/0.9780	**30.17/0.9335**	**25.09/0.8584**	34.71/0.9756	34.99/0.9683
BMBC	**39.90/0.9903**	35.31/0.9774	29.33/0.9270	23.92/0.8432	**35.01/0.9764**	**35.15/0.9689**
TOFlow	39.08/0.9890	34.39/0.9740	28.44/0.9183	23.39/0.8310	33.73/0.9682	34.58/0.9667
SuperSlomo	37.28 / 0.9861	33.80 / 0.9731	28.98 / 0.9250	24.15 / 0.8451	33.15 / 0.9664	34.75 / 0.9670
DVF	25.10 / 0.8480	23.31 / 0.8090	21.68 / 0.7681	19.86 / 0.7205	31.54 / 0.9467	34.12 / 0.9630
CAIN (original)	39.78 / 0.9901	**35.49 / 0.9770**	29.86 / 0.9290	24.69 / 0.8503	34.65 / 0.9738	34.91 / 0.9679
CAIN (distilled)	39.47/0.9893	34.96/ 0.9749	29.48/0.9235	24.57/0.8452	33.86/0.9700	34.79/0.9672

TABLE 5.3: Distilled CAIN compared with state-of-art models over different datasets

model	Time(s)
	1280×720
DAIN	1.033
BMBC	3.845
TOFlow	0.152
SuperSlomo	0.113
DVF	0.145
CAIN (original)	0.069
CAIN (distilled)	0.014

TABLE 5.4: Distilled CAIN compared with state-of-art models on 1280×720 resolution images

5.1.2 Qualitative Evaluation

FIGURE 5.1: Ground Truth and Overlaid Inputs

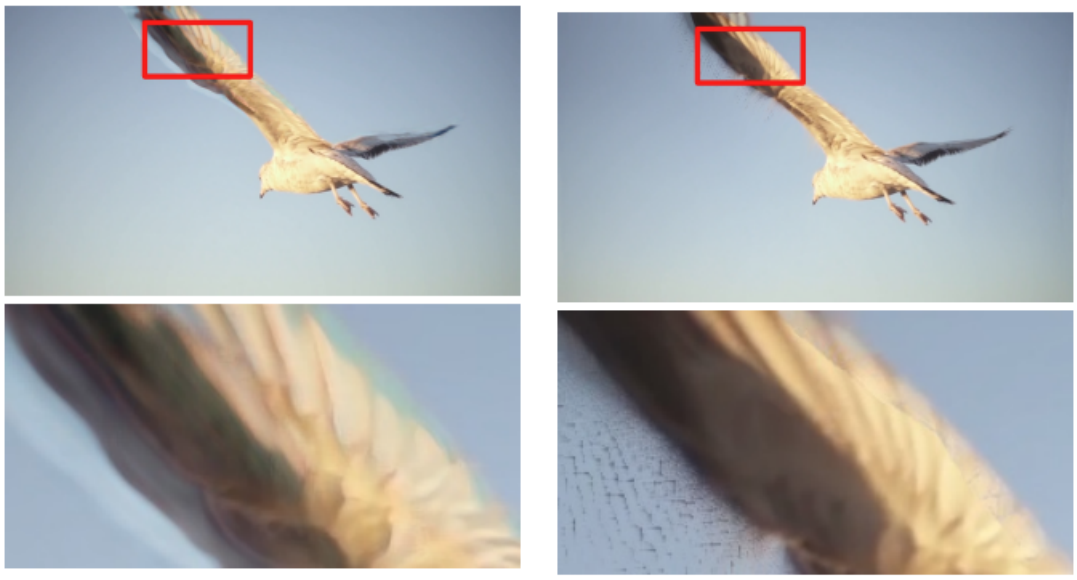


FIGURE 5.2: CDFI(left) and DAIN(right)

FIGURE 5.3: CAIN(left) and CAIN distilled(right)

The above images show a comparison of the distilled student model with other models on the dataset SNU-FILM (Choi et al., 2020). It can be seen that almost all state-of-art models produce artifacts, only varying in degrees.

5.2 Ablation Experiments

TABLE 5.5: Accuracy comparison on different distillation position settings on Vimeo90K dataset

	second last block + last block + head conv + tail conv	**last block + head conv + tail conv**	**last block + head conv**	**last block + tail conv**	**last block**
psnr	33.736	33.863	33.467	33.779	30.42483
ssim	0.963913	0.966961	0.960339	0.964461	0.897844

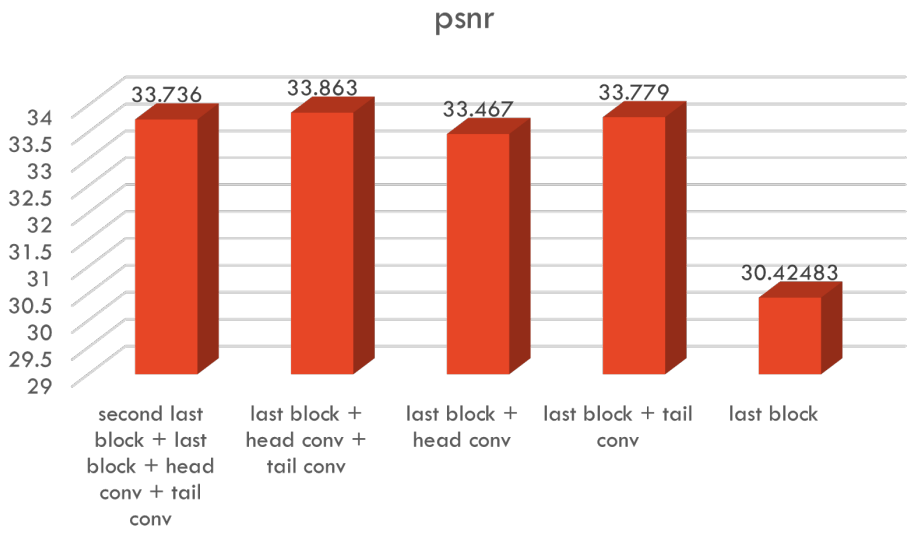


FIGURE 5.4: PNSR comparison on different distillation position settings

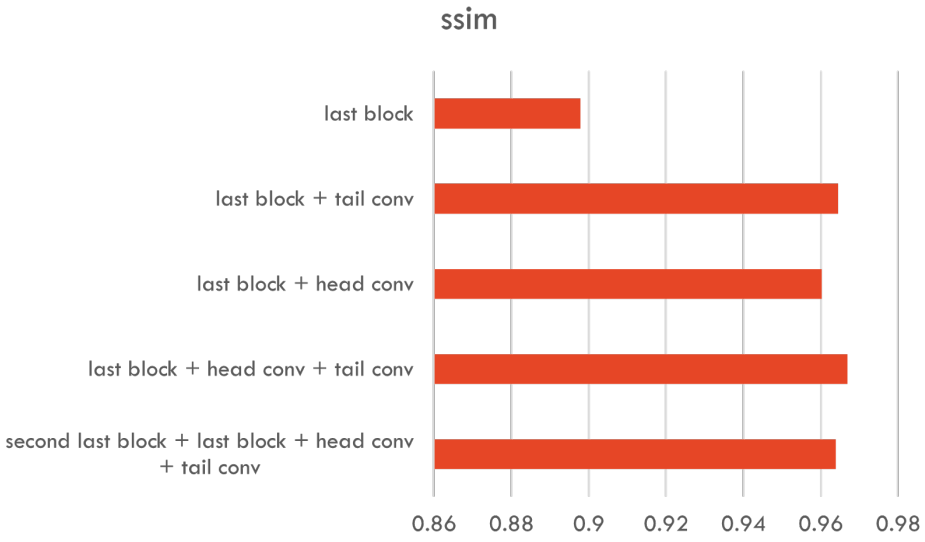


FIGURE 5.5: SSIM comparison on different distillation position settings

Changing the position and quantity of distillation sites in Figure 5.4 and Figure 5.5 reveals the effectiveness of the model after distillation. Based on the comparison of the first two columns in Figure 5.4 and Figure 5.5, it can be concluded that the number of distillation points is not optimal and that the outcome is enhanced by removing the distillation point in the second-to-last block of each group.

According to the Table 5.5, distillation at the final block and tail convolutional layer is more accurate than distillation at the final block and head convolutional layer. This verifies the notion that the distillation portion can get more information the closer it is to the output layer of the network.

TABLE 5.6: Different student structures performance on the Vimeo90k dataset

	1-block	**2-block**	**3-block**	**6-block**
psnr	32.622	33.38	33.863	34.293
ssim	0.952075	0.960234	0.966961	0.969336
parameter number(M)	5.5	9.3	12.6	22.7

Table 5.6 examines the outcomes generated from various student model designs when distilling from the final blocks, head convolutional layer, and tail convolutional layer. Even if just one block per group is kept, it is still possible to produce superior results through knowledge distillation. Depending on the needs of industrial applications, the number of layers of the student model can be altered to achieve a balance between precision and model size.

CHAPTER 6

Conclusion

Within the context of the study of video interpolation, this experiment investigates the impact that feature distillation can have. Channel Attention for frame Interpolation (CAIN) (Choi et al., 2020) is a well-established model in the field of video interpolation and is selected to serve as the primary support backbone. An efficient real-time student model is obtained as a result of the experiment. The student model inherits the accuracy of Channel Attention for frame Interpolation (CAIN) (Choi et al., 2020), is comparable to other models of state-of-art, and achieves optimal inference time and memory space, exceeding most state-of-art models.

It has been demonstrated through this experiment that the concluding feature distillation has the potential to enhance the performance of video frame interpolation in terms of inference time and memory space, making it more amenable to the requirements of industrial applications.

In the experiments described in this work that involve ablation, we continue our investigation into the properties of feature distillation and draw the following conclusions about our findings: 1. setting distillation points closer to the end layer will produce better results than setting them closer to the starting layer; 2. the number of distillation points needs to be accurately determined; 3. the loss that is generated from the teacher models needs to be given more weight than the loss that is generated from the ground truth.

Feature distillation may have other possible applications in video interpolation models, which might be investigated in further studies. Since the structure of the student model is not constrained in any way by the knowledge distillation process, it is feasible to investigate scenarios in which the teacher model and the student model each have their own distinct architectures. For example, a flow-based video interpolation model could be utilized by the teacher model, whilst a kernel-based model is utilized by the student model. Stacking numerous models as the teacher model is another approach that can be used

to investigate whether or not it is possible to improve the accuracy of the predictions made by the student model.

Bibliography

Romero Adriana, Ballas Nicolas, K Samira Ebrahimi, Chassang Antoine, Gatta Carlo, and B Yoshua. 2015. Fitnets: Hints for thin deep nets. *Proc. ICLR*, 2.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3703–3712.

Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802.

Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. 2020. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10663–10671.

Bucila Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *ACM SIGKDD*.

Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. 2021. Cdfi: Compression-driven network design for frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8001–8011.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Berthold KP Horn and Brian G Schunck. 1981. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2020. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*.

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9000–9008.

Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. 2020. Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv preprint arXiv:2012.08512*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Haopeng Li, Yuan Yuan, and Qi Wang. 2020. Video frame interpolation via residue refinement. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2613–2617. IEEE.

Zhengqi Li and Noah Snavely. 2018. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. 2017. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE international conference on computer vision*, pages 4463–4471.

Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. 2017. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891.

Simon Niklaus and Feng Liu. 2020. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446.

Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. 2020. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *European Conference on Computer Vision*, pages 109–125. Springer.

Junheum Park, Chul Lee, and Chang-Su Kim. 2021. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14539–14548.

Mary Phuong and Christoph H Lampert. 2019. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364.

Anurag Ranjan and Michael J Black. 2017. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402.

Suraj Srinivas and François Fleuret. 2018. Knowledge transfer with jacobian matching. In *International Conference on Machine Learning*, pages 4723–4731. PMLR.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR*, abs/1707.02968.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer.

Frederick Tung and Greg Mori. 2019. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1365–1374.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698.

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125.

Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L Yuille. 2019. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5628–5635.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4133–4141.

Sergey Zagoruyko and Nikos Komodakis. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.