POLITECNICO DI TORINO

Master Degree in Data Science and Engineering

Master Thesis

Sequential Domain Generalisation for Egocentric Action Recognition

Supervisor
Prof. Giuseppe Averta

Co-supervisors:
Dott.ssa Chiara Plizzari
Dott. Simone Alberto Peirone
Dott. Marco Ciccone

Candidate
Amirshayan Nasirimajd

April 2024

Sequential Domain Generalisation for Egocentric Action Recognition
Master thesis. Politecnico di Torino, Turin.

© Amirshayan Nasirimajd. All rights reserved.
March 2024.

The work in this thesis has been submitted for publication in IEEE Robotics and Automation Letters (RA-L).

Abstract

Due to the widespread popularity and accessibility of wearable devices, a substantial volume of egocentric (first-person) video data has become readily accessible. This has resulted in a growing interest of researchers in the field of egocentric vision understanding. This field of study holds significant potential in several areas, especially in robotics and the analysis of human behaviour. Indeed, gaining insights into human behaviour from an egocentric perspective can offer valuable insights to robotics experts, facilitating the development of robots with more human-like visual capabilities and a deeper comprehension of their surroundings similar to humans.

One of the main applications of egocentric vision is recognising the activities carried out by the wearer. However, one limitation when deploying action recognition models to real-world scenarios is that visual appearance data such as RGB inputs vary a lot when presented with new data distributions different from the training set, which inevitably leads to a decline in model performance. This issue is commonly known as the domain shift. Extensive efforts have been performed to increase model robustness across diverse domains (domain generalization).

To tackle this problem, we present an action recognition method that relies on actions' temporal context. In addition, to capture the actions' temporal context we decided to use the action sequences. The rationale behind this is that sequences of actions do not depend on the layout or appearance of the environment. Exploiting action sequences, we aim to produce a more generalized model and mitigate the adverse impact of domain shift.

In this thesis, we present Sequential Domain Generalisation (**SeqDG**), which is a reconstruction-based architecture to improve the generalization of action recognition models. This is accomplished through the utilization of a language model and a dual encoder-decoder that refines the feature representation. The model is trained with a visual-text sequence reconstruction objective (**SeqRec**) that utilises contextual information from both text and visual modalities to reconstruct a sequence's central action. Furthermore, we introduce **SeqMix**, a technique that mixes actions that share the same label but come from different domains to make the model more robust to visual changes.

We evaluate our approach's effectiveness and benefits in domain generalization on the EPIC-KITCHENS [1] and EGTEA [2] dataset. Our model is trained on a set of environments and tested on new unseen environments, showing the generalization benefits of the proposed approach. Extensive experiments show that our method improves by up to +2.4% compared to the baseline. This evidence suggests the proposed method's ability to improve model robustness and generalisation during domain shifts.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Artificial Intelligence (AI) stands at the forefront of some of the most significant developments in the field of computer science. The concept of simulating human learning capabilities through observation has become a foundational principle in the advancement of AI. This is achieved by the advancement in machine learning a subfield of AI, which allows machines to acquire the capacity to overcome and solve challenges by learning from available examples (data). In recent years, a large increase in accessible data and the growing processing power of GPUs have boosted the adoption of artificial neural networks as a learning method and the popularity of deep learning methods.

Deep learning illustrated its ability to solve complex tasks, and this resulted in the popularity of deep learning in the computer vision field. Within this field, many classical issues have been successfully addressed through deep learning techniques, such as object detection, object classification, and segmentation. Initially, these advancements were concentrated on image data before progressing to more complicated and challenging data, including videos. Consequently, the integration of deep learning techniques into video understanding tasks has forged a new trajectory and has given rise to novel research inquiries, such as robotic, and autonomous vehicle production.

Compared to images, videos introduce a greater level of complexity due to their inclusion of temporal dimension information, motion patterns, and increased dimensionality. For instance, deducing the ongoing activities in a video cannot be achieved through a single frame; it necessitates the incorporation of temporal information spanning the entire video duration. Numerous studies have undertaken the task of enhancing video comprehension by employing spatial-temporal data blocks for video analysis, such as the utilization of 3D convolutional layers [5, 15, 16], or some other approaches that aim to incorporate a lighter of temporal information through 2D convolutional layers [7, 17, 18, 19].

Furthermore, an additional issue within video understanding involves the matter of annotations and labels. Annotating video data is a resource-intensive process

Figure 1.1: Egocentric data sample gathered across different activities, and from people in different places in the world that wearing camera gadgets, taken from the [3].

that is time and cost-consuming. As a consequence, the diversity of video data tends to be lower in comparison to image data. This problem introduces the possibility that the training data distributions do not represent enough amount of data distributions. Therefore, Any model trained for a specific task might experience a decline in its exceptional performance due to the disparities between the training and testing datasets. Such a situation arises when the testing dataset introduces a completely new distribution of data to the model. This shift in data distribution between the training and testing phases results in what is known as a domain shift. Consequently, this discrepancy in domains leads to a reduction in the model's performance. As a result, it becomes vital to address this concern by maximizing the utilization of available data and enhancing the model's ability to adapt to new domains. Commonly this is done by using unlabeled data from the new domain (target domain), which this process is known as the unsupervised domain adaptation (UDA). However, the availability of unlabeled data from the target domain is not always guaranteed. Therefore, we need to maximize the model generalization by the data from the available domain for training. This approach is normally referred to as Domain Generalization (DG).

In terms of video understanding, the introduction of wearable gadgets like GoPros, or Google Glass has made a new type of data readily available: first-person view (egocentric) recordings, such as sample data available in Fig. 1.1. This data

is particularly interesting for researchers because, unlike the majority of videos captured in third-person view, it offers a window into human behaviour from the wearer's perspective. By studying egocentric data, we can gain valuable insights into how humans perform tasks and interact with their environment. This knowledge can then be applied to improve robots, allowing them to better mimic human-like actions and decision-making processes.

However, while video understanding is a complex task it can become more challenging if it is represented in an egocentric view, due to some specific characteristics of egocentric videos. In these videos camera normally is mounted on the head of the person, and close to the scene, as a result, the field of view can become narrow, and as a result, there will be not a complete representation of the environment. Furthermore, the constant movement of the camera wearer will represent the dynamic change in visual information which makes the understanding of the video even more complex for the machine. In addition, the lack of availability of diverse datasets is even more for egocentric data than other types of video data.

Consequently, the lack of data variation, along with additional egocentric video challenges amplifies the impact of domain shift on egocentric data compared to other types of video data. As a result, there has been extensive research on facing the domain shift problem for egocentric data [12, 14, 20, 13].

Nevertheless, most of these works are focused on a single action process of egocentric data, while one of the characteristics of egocentric data is the relations of the continuous actions that form the context of the actions. This context is because of the wearer's goal which results in a sequence of related actions that have been done by the wearer to fulfil his/her goal. In addition, the similarity between the goals in different domains results in a similarity between the sequence of actions in different domains. Therefore, the context in videos is domain-agnostic, and it can be explored as a solution for domain shift.

consequently, this thesis focuses on the use of sequences as a solution for domain generalization of egocentric action recognition. Moreover, by looking at the repeated sequences we try to learn the context and use them to make robust predictions.

1.1 Research goal and contributions

This work focuses on investigating the context effect on the robust understanding of long-term egocentric videos across different domains. In addition, we highlight this effect by assessing the similarity of sequences in different domains. knowing that human interaction is according to its goal and similar goals result in a repeated pattern within different environments. Therefore, we tried to take advantage of this phenomenon to provide a novel approach to generalize the model and mitigate the domain shift effect.

Consequently, we propose SeqDG, a novel approach for domain generalization in egocentric action recognition. SeqDG extracts the context of the action sequence and introduces a visual-text reconstruction objective. Here, for each modality (visual or textual), the model reconstructs actions masked in the sequence by using information from the opposite modality. Additionally, to optimize the model's ability to extract domain-agnostic features, we introduce sequences containing a mix of different actions across different domains.

Finally, we prove our approach performance in domain generalization by assessing our model in several well-known benchmarks. We analyse our proposed model on the EPIC-KITCHENS-100 [1] and EGTEA [2] benchmarks. The results indicated by SOTA performance illustrate the model's effectiveness in unseen environments in egocentric action recognition by using the sequence, and language model at the same time.

Chapter 2

Related Works

This chapter lays the groundwork for the thesis by introducing the background information and related works essential to understanding the research. Moreover, We begin by providing a foundation in core concepts in section 2.1. This section equips you with the necessary background knowledge. Subsequently, sections 2.2 and 2.3 delve deeper into more advanced concepts.

Also, this Chapter presents a comprehensive overview of some of the key existing research conducted in this field. We divide the work that has been done into two groups. First, we will start to introduce well-known models in action recognition and video understanding in section 2.4.4. Then, in section **??**, we will focus on some of the common practices in the field of domain generalization and adaptation.

2.1 Introductory Concepts

This section will explain the general concepts of deep learning, which are relevant to the work presented in this thesis. First, we will delve into the fundamental principles of machine learning in subsection 2.1.1. This will provide a solid foundation for understanding deep learning concepts. Then, we will focus specifically on deep learning primary concepts in the remainder of the section. Here, we will explore core concepts like perceptrons, activation functions, etc.

2.1.1 Machine Learning and Deep Learning

Machine learning (ML) and deep learning (DL) [21] are two powerful tools rapidly transforming various aspects of our lives. While both fall under the umbrella of artificial intelligence (AI). The concept of Machine Learning is about empowering computers to learn and improve without the need for explicit coding. Moreover, ML algorithms can learn a task by analyzing vast amounts of data, ML algorithms can identify patterns and relationships, enabling them to make predictions or decisions on unseen data [22].

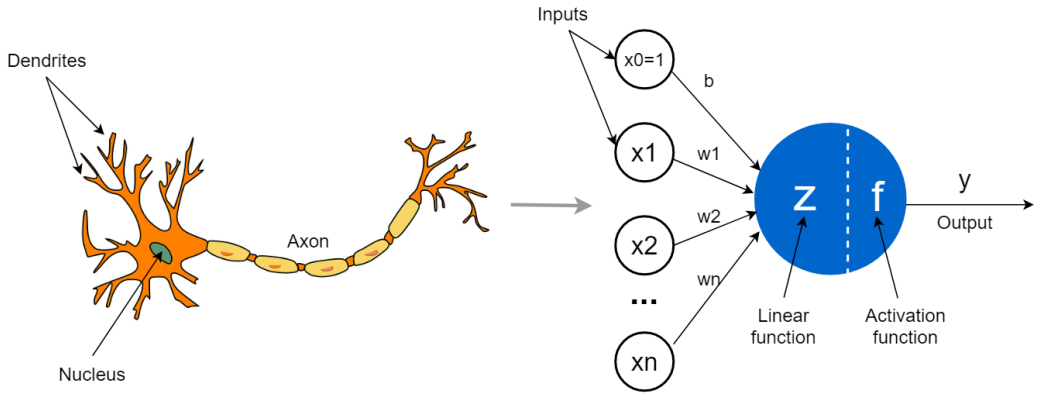


Figure 2.1: Perceptrons are simplified models of artificial neurons, inspired by the structure and function of biological neurons in the human brain, The following figure is taken from [4].

Furthermore, DL is a subfield of ML where its algorithm is inspired by the human brain function. In addition, DL models can learn complex tasks by introducing artificial neural networks (ANNs), which mimic the human brain neurons in learning, where a complicated task is distributed over many neurons, and each neuron is only responsible for doing a simple function. [21]

2.1.2 Perceptrons

The idea of the human neurons inspired the artificial neural networks. In this idea, each neuron will do a simple process and forward it to the next one. The simplest artificial neuron is called a perceptron [23], which is a simple binary linear classification function. Moreover, it receives a set of inputs and performs a weighted sum function to determine its output. It can be broken down into three main components:

- **Inputs**: Represented by a vector $X = (x_1, x_2, \ldots, x_n)$, where n is the number of input features.
- **Weights**: Represented by another vector $W = (w_1, w_2, \ldots, w_n)$, where each weight w_i corresponds to the importance of the i^{th} input.
- **Bias**: A constant term b added to the weighted sum, allowing the perceptron to shift the decision boundary.

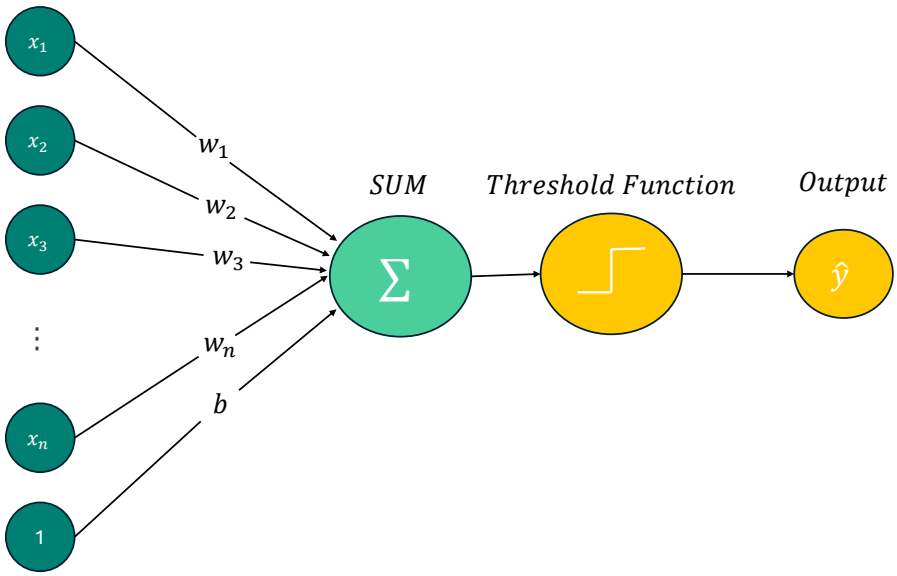


Figure 2.2: A perceptron unit as a simple function used in artificial neural networks.

The perceptron's functionality can be formulated mathematically. The weighted sum of the inputs is calculated as:

$$Z = \sum_{i=1}^{n} w_i x_i + b \tag{2.1}$$

This value, Z, is then passed through a threshold function, denoted by σ as the step function:

$$\sigma(Z) = \begin{cases} 1 & \text{if } Z \geq 0 \\ 0 & \text{if } Z < 0 \end{cases} \tag{2.2}$$

The step function classifies the input as belonging to one class (output of 1) or another class (output of 0) based on a linear decision boundary.

2.1.3 Multi-Layer Perceptrons

Multi-layer perceptrons (MLPs)[24, 25] represent a complex mimicking of the human neural networks compared to single perceptron, and they form the foundational architecture in the artificial neural networks (ANNs), offering a complex framework for solving machine learning tasks. They consist of multiple layers of interconnected nodes (neurons). As indicated in Fig. 2.3 the layers typically include an input layer to receive data, one or more hidden layers responsible for capturing complex patterns within the input, and an output layer to produce the desired predictions

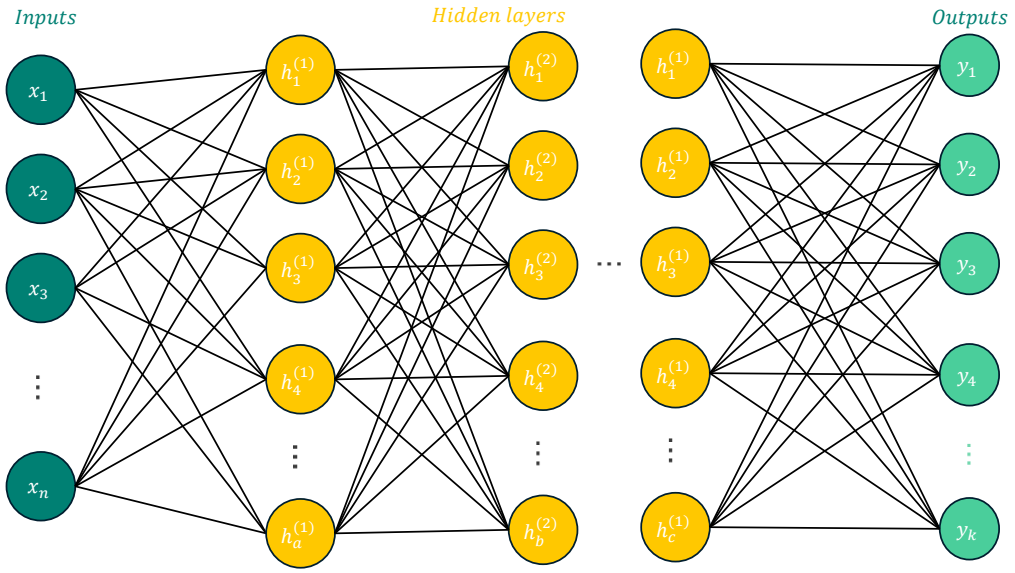


Figure 2.3: A Multi-Layer Perceptron (MLP) mimics human neural networks with n inputs, multiple hidden layers, and k outputs. Here the connection between each layer presents the weights that will determine the output of each neuron.

or classifications. Each connection between nodes is associated with a weight w_i. Through a process known as backpropagation 2.1.6, MLPs adjust these weights during training to minimize the discrepancy between predicted and actual outputs, thereby enhancing their ability to learn the final task.

The process for the i-th layer within a Multi-Layer Perceptron (MLP) can be mathematically represented using the following formula:

$$\hat{O}_i = \phi(\hat{O}_{i-1}.W_{i-1,i} + b_i) \tag{2.3}$$

Where the $\hat{O}_i$ is the output of the layer i, and the $W_{i-1,i}$ is the weight matrix between layer i and $i-1$, b is the bias, ϕ is the activation 2.1.4 function, and the $\hat{O}_{i-1}$ is the output of the previous layer. Moreover, the activation function 2.1.4 ϕ is present to introduce non-linearity into the model, allowing it to learn complex relationships between the inputs and the output. Finally, The process of feeding the first layer with the inputs, and forwarding the output of each layer to the next layer, until the final layer produces the output $\hat{y}$ is called the feedforward process, and it is the main function of the MLPs to produce their results.

2.1.4 Activation Functions

MLPs are powerful tools for tackling complex problems one of the important components of MLPs is activation functions. They take the weighted sum of inputs from

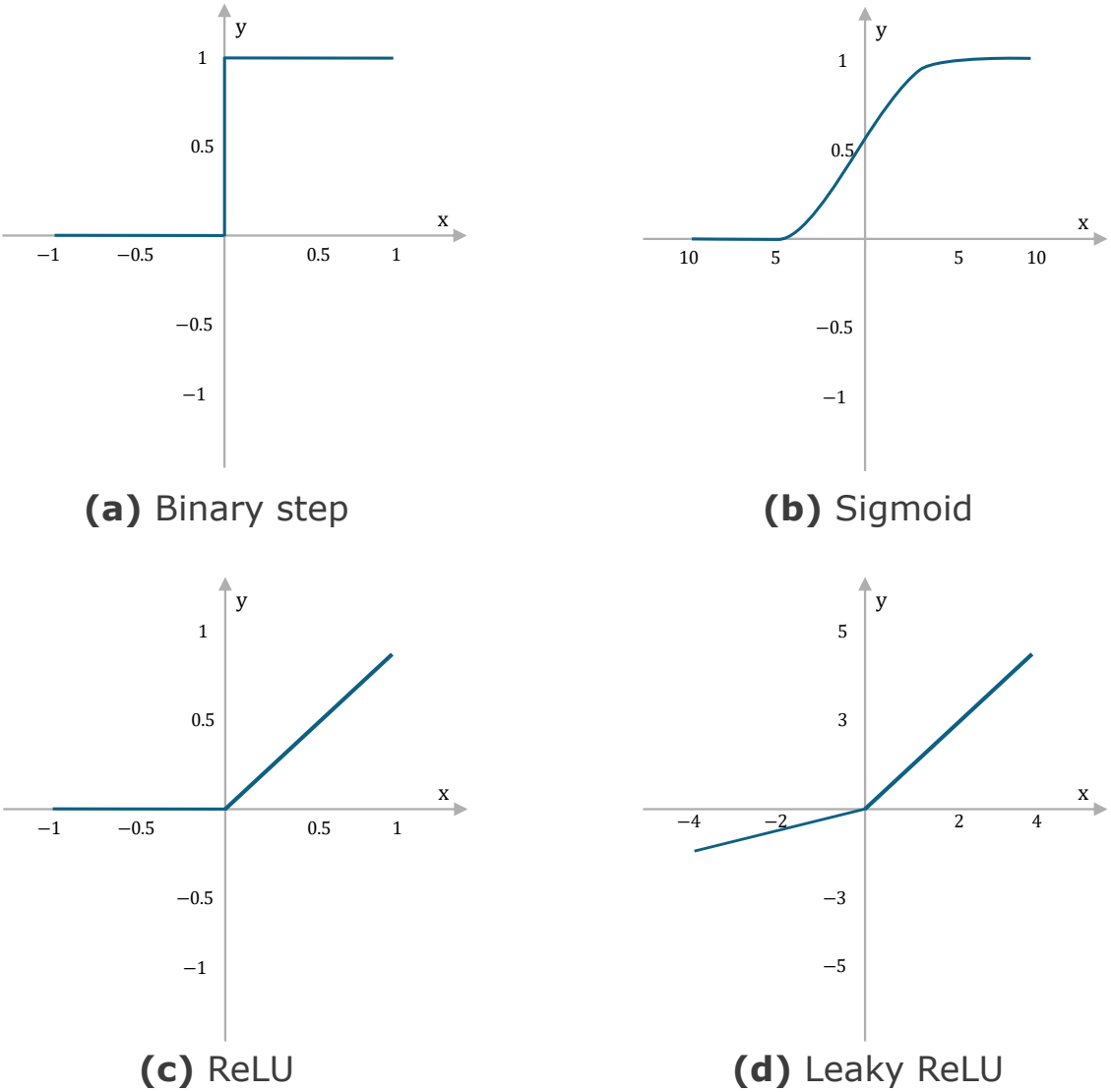


Figure 2.4: The activation functions plotting.

previous layers, often along with a bias term, and apply a non-linear transformation to determine the neuron's output. This is done to avoid linear transformations, limiting the ability of MLPs to learn complex patterns.

Moreover, the activation functions affect the learning process of models, and to better understand them we will introduce some of the well-known activation functions as indicated in Fig. 2.4. Additionally, the choice of function depends on the specific problem and network architecture. Understanding these core functions empowers you to unlock the true potential of MLPs and build powerful machine-learning models.

Binary Step Function

The binary step function is a threshold-based classifier. It is one of the simplest activation functions, and it decides whether to activate the neuron or not. In addition, it is Mathematically expressed as:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

This function outputs a 1 (activated) if the input (x) is greater than or equal to 0, and a 0 (deactivated) otherwise. While conceptually simple, its limitations, like not being differentiable which is needed for the learning process as described in the subsection 2.1.6, and restrict its use in modern MLPs.

Sigmoid Function

Besides the binary step function, a popular choice for non-linear activation in artificial neural networks is the sigmoid function. This function belongs to the category of logistic functions and offers a key advantage: its derivative can be easily calculated. This characteristic is crucial for training these networks. Regardless of the value fed into the sigmoid function, the output will always fall within a specific range – between 0 and 1. Described by the formula:

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{2.5}$$

This function ensures the output remains within a manageable range. However, its vanishing gradients in certain regions can hinder training in deep MLPs.

ReLU (Rectified Linear Unit)

The ReLU (Rectified Linear Unit) [26] activation function has become popular for its efficiency and effectiveness. It acts like a binary step function for negative values, but for positive input values, letting them pass through unchanged. Unlike some other activation functions, ReLU's output isn't confined to a specific range. Additionally, the positive half of its operating range boasts a constant, positive slope when we calculate its derivative, which is helpful for training purposes.In addition, it can defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

This function is computationally inexpensive and avoids the vanishing gradient problem (a commonly known problem in the deep learning process). However, ReLU can suffer from "dying ReLU" issues where neurons get stuck at zero.

Leaky ReLU

Leaky ReLU [26] is a variant of ReLU that introduces a small positive slope for negative inputs. Represented as:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.1x & \text{if} x \leq 0. \end{cases} \tag{2.7}$$

This allows a small gradient to flow even for negative inputs, potentially preventing neurons from getting stuck and improving training.

2.1.5 Loss Functions

To better understand a model's performance, we quantify the discrepancy between an artificial neural network's predicted outputs ($\hat{y}$) and the actual, desired outputs (y). This numerical value, crucial in supervised learning frameworks, serves as a guide during the training process. Therefore, the objective becomes minimising the loss function, prompting the network to adjust its internal parameters (weights and biases) to progressively improve its predictive accuracy. Common loss functions include **mean squared error (MSE)** for regression tasks and **cross-entropy** for classification problems. The choice of loss function depends on the specific problem and the nature of the output variable.

Mean Squared Error (MSE)

The mean squared error (MSE) loss function is a common loss function for evaluating a model's performance in regression tasks. It calculates the average squared difference between the predicted outputs ($\hat{y}$) of a model and the ground truth values (y) in the training data. Mathematically, MSE is expressed as:

$$MSE(\hat{y}, y) = \frac{1}{N} \times \Sigma(y_i - \hat{y}_i)^2 \tag{2.8}$$

where N represents the total number of data points and the summation iterates over all data points (i). This formulation essentially squares the errors for each prediction, penalizing larger deviations more heavily. By minimizing the MSE loss function during training, the model learns to adjust its internal parameters to produce predictions that closely align with the actual values, resulting in improved regression accuracy. MSE is particularly suitable for problems where the magnitude of errors is crucial, as squaring the errors emphasizes larger discrepancies.

Cross-Entropy Loss

The cross-entropy loss[27, 28] function is fundamental in classification tasks. In addition, it measures the difference between the probability distributions of an

artificial neural network's predictions ($\hat{y}$) and the true labels (y). Unlike the mean squared error function used in regression, cross-entropy focuses on the information content of these distributions. Mathematically, it's often formulated as:

$$CrossEntropy(\hat{y}, y) = -\Sigma(y_i log \hat{y}_i) \tag{2.9}$$

2.1.6 Learning and Optimization

Considering the objective of the ANN models to be able to map the inputs to the desired output, and minimize the discrepancy between the predicted and actual outputs (loss function), the ANN models go through a gradual learning process. In this process, we iteratively reduce the distance between the output and the desired output. The method that is commonly used for this goal is called gradient descent (GD), an optimization method that focuses on minimizing the loss function iteratively. The GD method starts at a random point and uses the negative of the slope, which is the gradient of the function, to determine the direction of the minimum point. Then, by using a hyperparameter called the learning rate, it determines the magnitude of the move towards the minimum point. Additionally, the gradual movement is repeated until the minimum is found or a pre-defined stopping criterion is met, such as reaching a maximum number of iterations or achieving a sufficiently low loss value.

Gradient Descent and Backpropagation

Let's delve deeper into the learning process by exploring the math behind it. To optimize the network using Gradient Descent (GD) [29, 30], we need to determine the direction for updating each weight. This is achieved by calculating the derivative of the loss function, starting from the output layer and working our way backwards to the first layer. This calculation process is known as backpropagation [31, 32].

To better understand let's consider that the $J(\theta)$ is the objective function, where we focus on minimizing it according to the parameter θ. Therefore, we can define our problem as follows:

$$\begin{aligned} \theta^* &= arg_\theta min J(\theta), \\ J(\theta) = E_{(x,y)\sim p(x,y)} L(x, y; \theta) &= \frac{1}{N} \sum_{i=1}^{N} L(x^i, y^i; \theta), \end{aligned} \tag{2.10}$$

Where here we assumed the objective function is the mean of the loss function L. Moreover, the objective function $J(\theta)$ is also called the empirical risk of the model. Having two sets of samples $p(x, y)$ as the training sample set, and $p^{\sim}(x, y)$ as the test set, we use the name empirical to indicate the difference between these sample sets.

In addition, to achieve the best θ parameter we update its value over time using GD method at each time step t. However, we can use GD if and only if the objective function is convex. As a result, the derivative of the objective function at time t is calculated as $\nabla J(x, y; \theta^t)$, and by having the derivative the θ^{t+1} is calculated as follows:

$$\theta^{t+1} = \theta^t - \gamma \nabla J(x, y; \theta^t) \tag{2.11}$$

Here γ is an indicator of the magnitude of change in our model, and it is called the learning step.

2.1.7 Regularization

One of the common problems in training Artificial Neural Networks (ANNs) is over-fitting. This occurs when the model becomes too focused on learning the specific patterns and noise present in the training data. As a result, the ANN excels at mapping the training data but performs poorly on unseen data. This is because the model has not learned to generalize the underlying relationships within the data, instead memorizing the training examples themselves. Therefore, there have been several methods presented in terms of increasing the model performance, and we will discuss some well-known methods.

Early Stop

One of the common methods for stopping over-fitting is an early stop. In this method, the training data is divided into two splits (training and validation). Therefore, during the training, the early stopping method continuously monitors the model's performance on a separate validation set as training progresses. Once the model's performance on the validation set starts to decline, indicating overfitting on the training data, training is stopped. This prevents the model from further memorizing irrelevant details and allows it to focus on generalizable knowledge, ultimately leading to better performance on unseen data.

Dropout

Another common known to avoid over-fitting is called dropout [33]. During each training epoch, dropout randomly disables a set percentage of neurons in a layer. These "dropped-out" neurons don't contribute to the calculations, forcing the network to learn redundancies. Instead of relying on specific connections, the network must adapt to function well with different active neurons each time. This discourages the network from overfitting to the training data and promotes learning more robust features that generalize better to unseen data. It's like training a diverse ensemble of slightly different networks within your main one, leading to improved performance on new information.

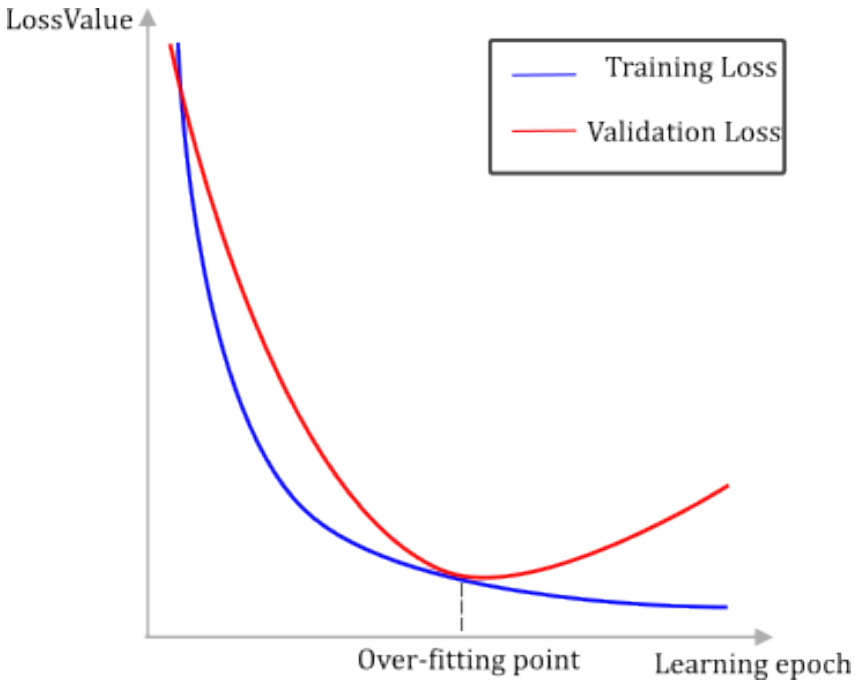


Figure 2.5: Here we can see the over-fitting point is the point where validation loss increases while the training still decreases.

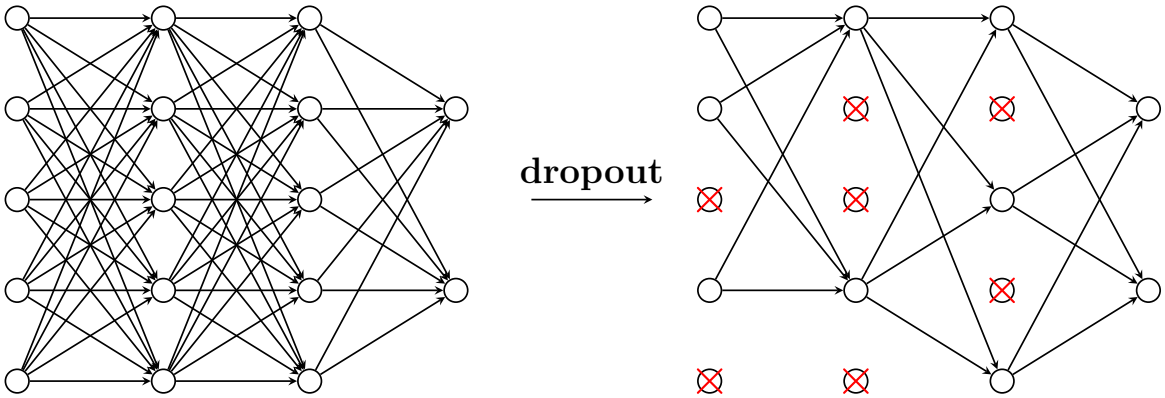


Figure 2.6: A sample dropout during training, where a group of nodes are randomly eliminated in an epoch of training, resulting in learning their task by other nodes and increasing robustness.

2.2 Convolutional Neural Network (CNN)

One of the important applications of ANNs is in computer vision. In recent years there has been a huge advancement in this area due to the introduction of Convolutional Neural Network (CNN). Convolutional Neural Networks (CNNs) [32, 34]

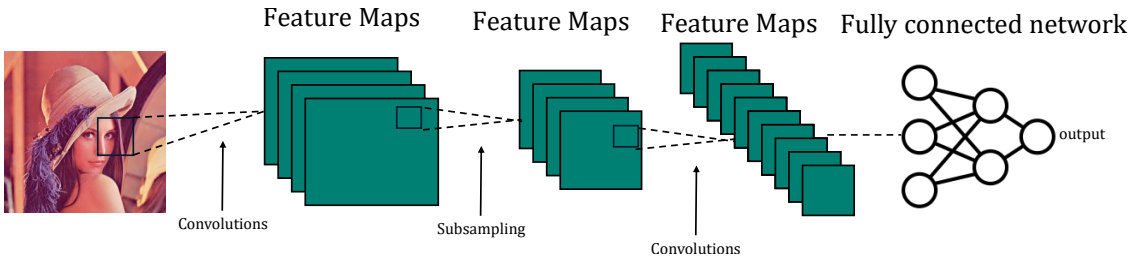


Figure 2.7: A sample of CNNs where image features are extracted step by step and then fed to a fully connected network.

is a powerful type of Artificial Neural Network (ANN) specifically designed to handle visual information for tasks such as image recognition and analysis tasks. In comparison to normal ANNs that treat images as flat arrays of input values, CNNs scan areas of the image to extract information in a grid-like fashion. This enables CNNs to extract meaningful features and patterns from images more efficiently than traditional ANNs.

Furthermore, The core building blocks of a CNN are the convolutional layer and pooling layer, Which will be discussed. First in the subsection 2.2.1, we will introduce the convolutional layer and then in the subsection 2.2.2 we will talk about pooling layers.

2.2.1 Convolutional Layer

Convolution is a well-known mathematics operation for signal processing, which indicates the interaction between two functions f and g, and it is defined as follows:

$$(f * g)(t) := \int_{\infty}^{-\infty} f(\tau)g(t-\tau)d\tau \tag{2.12}$$

In the case of discrete functions, the following equation will be used:

$$(f * g)(n) = sum_{n=-\infty}^{\infty} f[m]g[n-m] \tag{2.13}$$

However, the implementation of this method in computer vision is to create layers that apply a series of filters (also called kernels) that slide across the image, analyzing small regions at a time. Each filter detects specific features, like edges, lines, or shapes. As the filters move across the image, they generate activation maps, highlighting the presence of those features in different locations. By stacking multiple convolutional layers, CNNs can progressively build up more complex features from simpler ones, ultimately leading to object recognition.

As indicated in Fig. 2.8, we can see a kernel with a size of $n \times n$ moves along the image and create the feature grid.

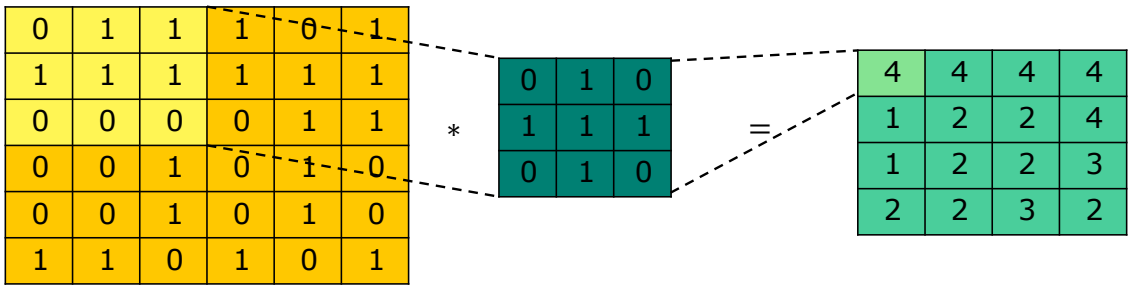


Figure 2.8: Convolutional function sample process.

2.2.2 Pooling layers

Beyond convolutional layers, CNNs often use pooling layers to downsample the activation maps. This is done to reduce feature dimensionality while trying to not lose important information. This helps to reduce the network's complexity and prevents overfitting. In addition, To reduce the spatial dimensions (width and height) of the feature maps generated by convolutional layers, pooling layers apply a filter (often a small square) that slides across the feature map with a certain stride (step size). For each position of the filter, a pooling operation is performed on the elements within the filter's boundaries. There are three main methods for pooling operations, which are max pooling, average pooling, and min pooling. However, the most common method is max pooling, where the highest value in the area of the filter will be sampled.

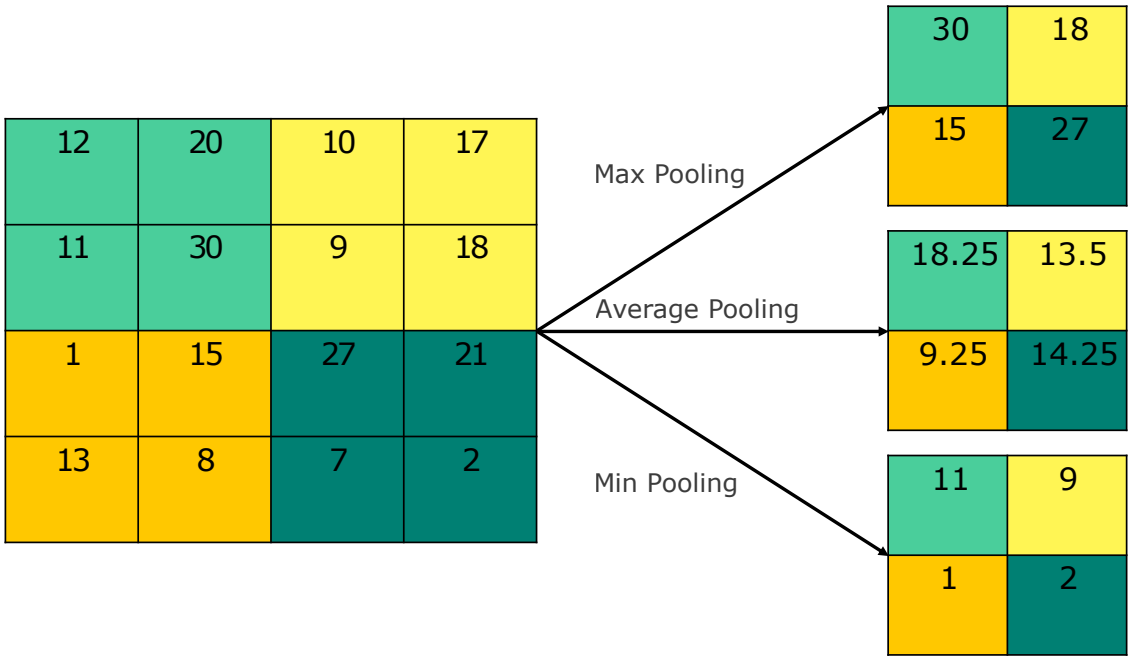


Figure 2.9: Different Pooling methods.

2.3 Sequential Data

While ANN models excel at uncovering hidden patterns in data, not all data is created equal. Moreover, some types of data find meaning over time or by being processed in an inherent order. These types of data are called sequential data, such as videos, Audio data, or Text data. This type of data presents unique challenges and opportunities with themselves and results in different approaches to unfold their information. Here, we will discuss some of the well-known approaches towards these types of data.

2.3.1 Recurrent Neural Network(RNN)

Recurrent Neural Networks (RNNs) [35] are a powerful type of ANN architecture designed specifically for handling sequential data. While traditional neural networks are built for static inputs, RNNs can perform tasks where the order of information matters. Therefore, They are the ideal type of models for applications in natural language processing (NLP), speech recognition, and time series forecasting.

The core of an RNN model is the concept of internal memory state. This state captures information from previous elements in the sequence and influences the processing of current and future elements. Mathematically, an RNN can be represented as follows:

$$[t]h(t) = f(U \times x(t) + W \times h(t-1)) \tag{2.14}$$

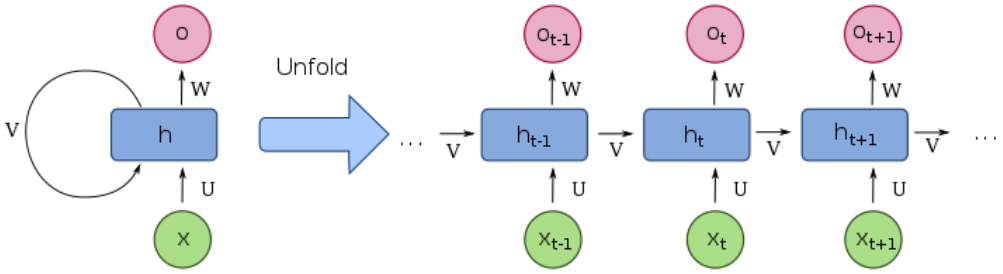


Figure 2.10: Recurrent Neural Networks (RNNs) where each stat $h(t)$ provides the output $O(t)$

Here the $h(t)$ represents the hidden state vector at time step t, which indicates the network's memory processed sequence so far. Then, the $x(t)$ is the input at the time step t, U and W represent the weights learned during training, and f is the activation function.

Furthermore, the equation presents how the hidden state is updated at each time step t, where the current input $(x(t))$ is multiplied by weight matrix U, and the previous hidden state $(h(t-1))$ is multiplied by weight matrix W. Then the sum of these products is passed through the activation function f to create the new hidden state $h(t)$. This process exists for the whole sequence length, allowing the RNN to progressively build its understanding based on the order of the information.

However, RNNs suffer from a major problem, which is the vanishing gradient problem. For long sequences, the effect of earlier elements in the sequence can diminish as the backpropagation algorithm updates the weights. This is where other methods are introduced such as Long Short-Term Memory (LSTM) networks, which are specifically designed to address this limitation and enable RNNs to effectively learn long-term dependencies within sequential data.

2.3.2 Long Short-Term Memory(LSTM)

The limitation in Recurrent Neural Networks (RNNs) such as the vanishing gradient problem, hinders their ability to learn long-term dependencies within time-dependent data. As a result, the Long Short-Term Memory (LSTM) [35] networks come in. LSTMs are a specific type of RNN architecture designed to address this limitation.

A gating mechanism introduced by LSTMs that controls the flow of information within the network. This allows LSTM's cells to selectively remember or forget information over time. While a standard RNN cell simply updates its hidden state based on the previous state and current input, LSTMs incorporate memory cells

and gates that regulate the flow of information. Therefore, LSTMs are able to learn complex relationships between elements in long sequences, where earlier elements might significantly influence later ones.

Mathematically, an LSTM unit can be represented by a series of equations involving gates and cell states:

$$\begin{aligned} f(t) &= \sigma(W_f * [x(t), h(t-1)]) \\ i(t) &= \sigma(W_i * [x(t), h(t-1)]) \\ \tilde{c}(t) &= tanh(W_c * [x(t), h(t-1)]) \\ C(t) &= f(t) * C(t-1) + i(t) * \tilde{c}(t) \\ o(t) &= \sigma(W_o * [x(t), h(t-1)]) \\ h(t) &= o(t) * tanh(C(t)) \end{aligned} \tag{2.15}$$

Here, σ indicates the sigmoid activation function. The $tanh$ shows the hyperbolic tangent activation function. W_f, W_i, W_c, and W_o are specific weights to each gate. These formulations define how the forget gate ($f(t)$), input gate ($i(t)$), candidate cell state ($\tilde{c}(t)$), cell state ($C(t)$), output gate ($o(t)$), and hidden state ($h(t)$) are calculated at each time step. The forget gate controls the amount of information, which is forgotten from the previous cell state, the input gate determines how much new information is incorporated, and the output gate regulates what information from the current cell state flows to the hidden state, which is ultimately used for prediction.

Finally, by utilizing these gates, LSTMs can effectively learn long-term dependencies within sequences, making them a powerful tool for various tasks involving sequential data. This capability has led to significant advancements in areas like machine translation, speech recognition, and time series forecasting.

2.3.3 Transformers

While RNN and LSTM gain huge advancement in working the sequential data tasks, they are suffering from different limitations such as slow training due to sequential processing, and not being able to completely capture long-range dependencies within sequences. These problems lead to the creation of transformers [36], a powerful alternative architecture addressing these disadvantages of traditional sequential ANN models.

Transformers introduced the attention mechanism as a substitute method to understand the relationship in sequence instead of sequential processing data one by one. Using the attention mechanism they can learn to put a weight focus on a specific part of specific parts of the input sequence that are most relevant for processing tasks, and they do this in parallel. This parallel processing power makes Transformers significantly faster to train compared to traditional methods.

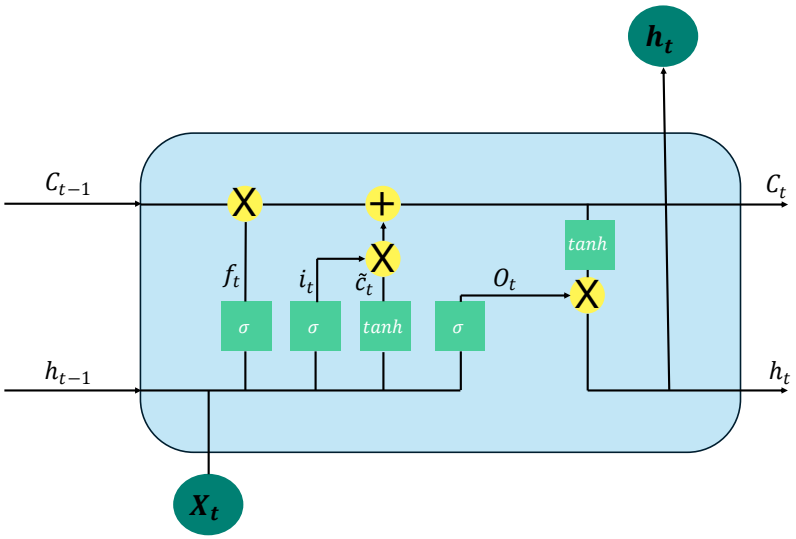


Figure 2.11: A Long Short-Term Memory(LSTM) cell.

The attention as the core of transformers follows the following equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) * V \tag{2.16}$$

Here Q represents the query, which is a vector encoding the current element being processed, and K is the key, vectors encoding all elements in the sequence. Then V indicates the value, which is also vectors that encode all elements in the sequence, and finally d_k denotes the dimensionality of the key vector.

Moreover, for each element in the sequence, this equation will calculate the attention weights using its relevance to the query (current element). Having these weights, we can create a context vector by summing the attention score base weighted values of all elements, this process is called encodings. As a result of incorporating information from relevant parts of the sequence, the context vector can illustrate a better understanding of the current element.

Furthermore, by stacking multiple self-attention layers, transformers can better fine-tune their understanding of the sequence and capture long-term dependencies. In addition, they were able to use the attention weights to introduce the encoder-decoder architecture as it has been shown in Fig. 2.12. This architecture creates an understanding of elements in an encoded vector space in the encoding phase, and by using the attention from the encoder they are able to recreate this understanding in a desirable form in the decoding phase. Therefore, transformers gain noticeable advances in different fields such as machine translation, and multiple models for encoder phase [37, 38, 39, 40] and decoder phase [41, 42, 43] of transformers introduced.

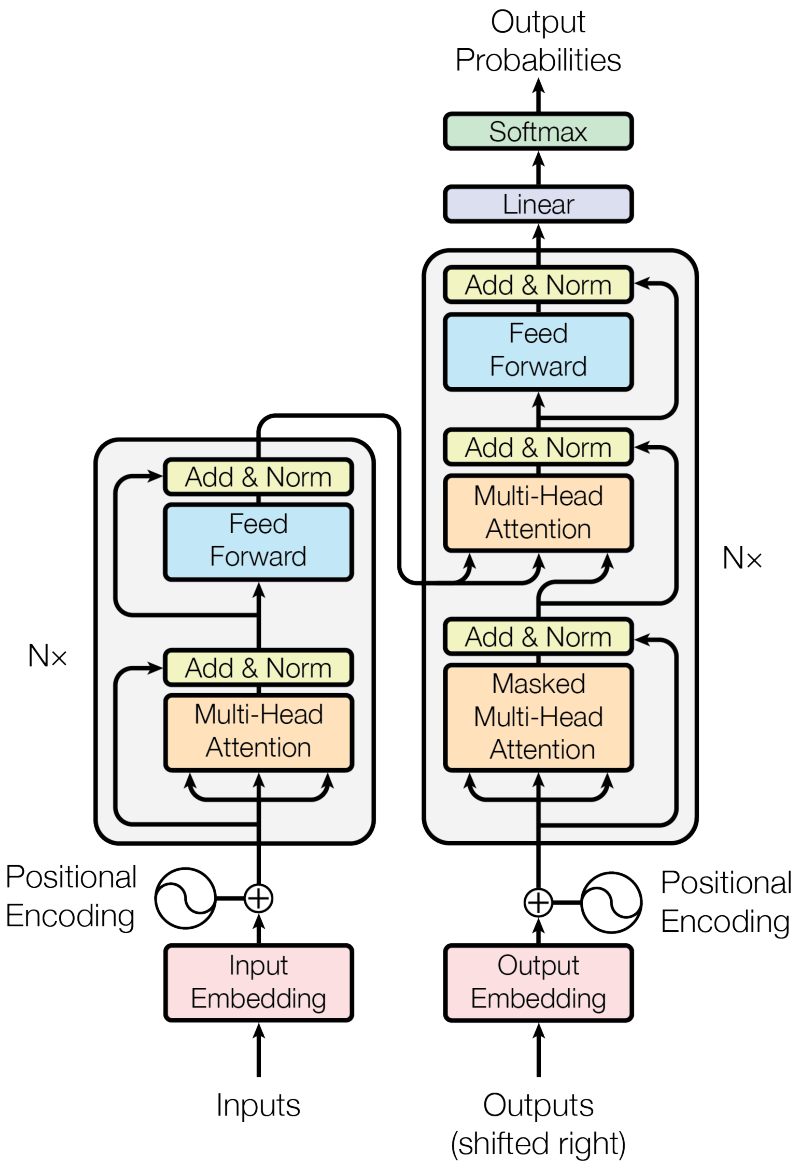


Figure 2.12: The encoder-decoder architecture of transformers.

2.4 Egocentric Action Recognition

Egocentric action recognition (EAR) is a subfield of video understanding that focuses on understanding and classifying human actions from the first-person view. This perspective is captured by wearable camera gadgets like head-mounted cameras or smart glasses, providing a view of the world similar to human perception. Unlike normal action recognition in third-person videos, egocentric action recognition suffers from several challenges, such as a limited field of view due to the closeness of the camera to the scene, or constant change in the scene due to the

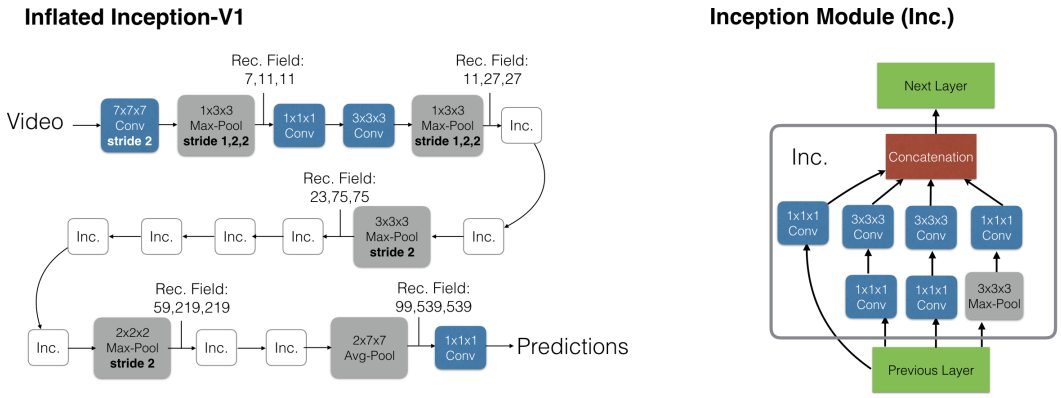


Figure 2.13: The left image is the Inflated Inception-V1 architecture and the right image is its detailed inception submodule the figure is taken from [5].

human motion. Therefore, multiple works tried to mitigate these effects and introduced EAR action recognition models[44, 45, 46, 47, 48].

In this section, we are going to introduce the common approaches to the EAR problem starting from 3D base approaches 2.4.1 and moving forward to the most recent ones.

2.4.1 3D Base Action Recognition

Egocentric action recognition models can be categorized into two categories 2D [49, 18, 7, 19, 8, 50, 51] base and 3D [12, 52, 16, 53, 15, 17], base models. In 3D base models Inflated 3D ConvNet (I3D) [5] is one of the well-known methods. This model was introduced to understand the use of the temporal context of videos by leveraging the 3D convolutional neural networks(3D CNNs). The 3D nature of the I3D model was able to extract both spatial and temporal features from video data. Therefore, the I3D is considered a well-suited architecture for recognizing human actions within videos.

As indicated in Fig. 2.13, the I3D is built by taking advantage of using the 2D ConveNets as the building blocks, specifically the Inception architecture [54]. The main idea of I3D was around the inflating of the 3D filters and pooling kernels into their 3D counterparts. Therefore, the I3D model was able to learn the spatio-temporal features by directly using raw video sequences. In this model, multiple layers of convolutional layers will reduce the dimensionality, using the pooled activations. To do this the 2D filters repeated N times along the time dimension to reduce the dimension by the factor of $1/N$. Additionally, The I3D model kept the two-stream architecture using the RGB, Optical Flow streams. This is done to use

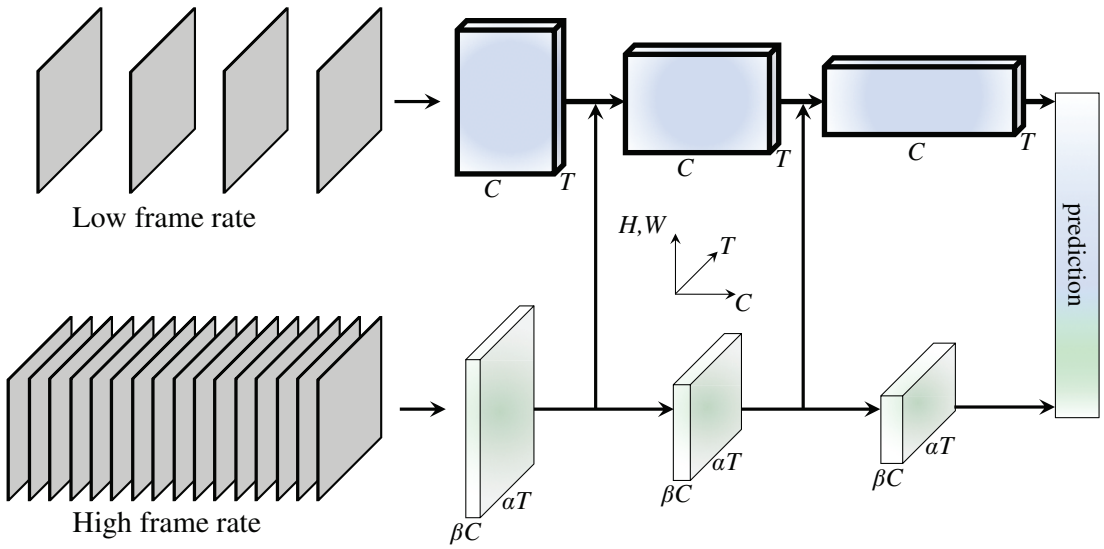


Figure 2.14: The SlowFast two stream illustration from the paper [6]. In this figure, the top sampling is the low frame rate for the slow pathway, and the lower stream is the high frame rate for fast pathway.

the benefits of the recurrent information that is not available in the RGB streams.

Another well-known 3D base model is the SlowFast[6]. The SlowFast model was introduced as a video understanding model with the idea of extracting the temporal information at different granularities. This model focuses on extracting information from two types of movement speed in video, the fast-paced micro-movements, and the slow overarching gestures. Having both types of movement pace information at the same time in the video enabled SlowFast to achieve a more comprehensive understanding of the video, while most traditional approaches lack the ability to capture the two movement speeds simultaneously. As a result of SlowFast's capabilities, it is considered one of the most common methods in the field of action recognition.

Looking at Fig. 2.14, SlowFast deployed a two-stream architecture for two different temporal resolutions. The two streams are called the slow pathway and the fast pathway. The slow pathway puts its attention on extracting information from the slower, semantic aspects of the video. To do so they use a low frame rate sampling in the video and investigate the large temporal contexts to identify overarching actions. On the other hand, the fast pathway focuses on high frame rate sampling to better capture information from faster, finer-grained motions within the video. In addition, the fast pathway uses the full temporal resolution of the video, enabling the model to assess rapid movements and subtle details.

Having both slow and fast pathways, the SlowFast [6] will fuse them to build a more comprehensive understanding of the video. After the fusion of the model,

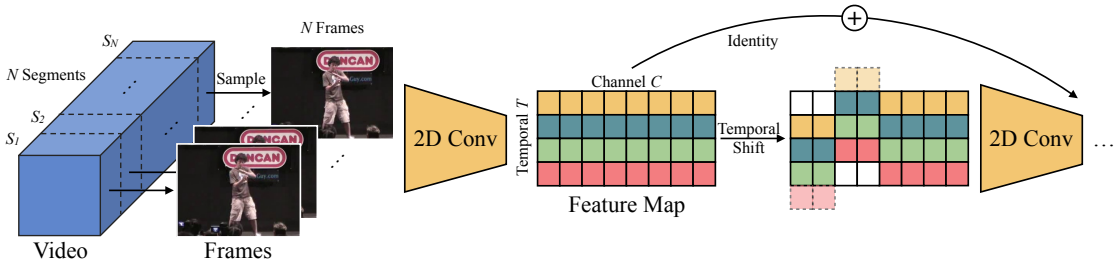


Figure 2.15: The TSM overall illustration from the paper [7].

we are able to capture both short-term and long-term temporal information within the video. This approach resulted in higher accuracy in action classification, and flexibility to adapt by adjusting sampling rate. .

2.4.2 2D Base Action Recognition

While 3D models were able to capture the time dimension, they lacked efficiency in terms of computation time and training. As a result, several works focused on using the 2D CNN models as an efficient substitute for the 3D models. The Temporal Shift Module (TSM) architecture [7] is one of the well-known models for using 2D CNNs in video understanding. The TSM was able to ta. The Author of [7] was able to tackle the challenge of capturing temporal relationships within videos without incurring significant computational overhead.

The TSM introduced a simple idea to integrate the information from different parts of the temporal context into one another. To do so, Author [7] use shuffling the feature across the temporal dimension. While traditional 2D CNNs can demonstrate great performance at capturing spatial features within a single video frame, they are suffering from the ability to capture crucial information about how these features evolve during the time dimension. As illustrated in Fig. 2.15, TSM tackles this limitation by shifting a subgroup of feature channels along the temporal axis from past and future frames with the current frame. As a result, the model learns to highlight the temporal dependencies.

2.4.3 Multimodal Egocentric Action Recognition

While most of the general action recognition architectures excel at action recognition, they still struggle to provide strong performance in egocentric videos due to the special characteristics of egocentric data such as a limited field of view, or sudden ego-motion. As a result, several methods focus on mitigating the effects of these features and provide a more robust method for egocentric action recognition. Temporal Binding Networks (TBN) [8] is a method specialized in egocentric vision by leveraging not only the visual information but also the audio information

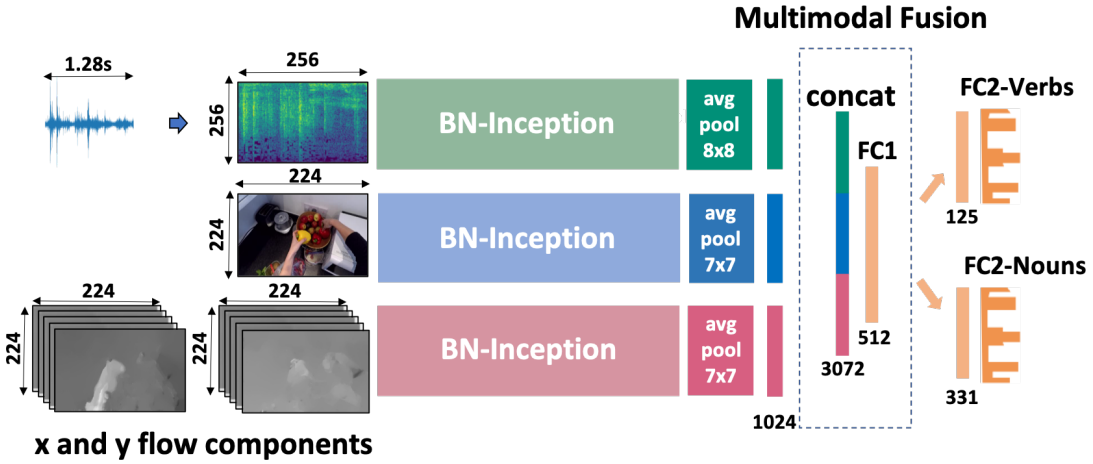


Figure 2.16: The TBN architecture in using three modalities, taken from the paper [8]

presented in the video.

Looking at Fig. 2.16, we can see The TBN uses three modalities (RGB, Flow, and Audio) for a more robust classification. The author of [8] focuses on the importance of sound as the indicator by indicating that the sound of chopping vegetables is as informative as visual data when recognizing the action of cooking. In addition, to use these modalities at the same time the TBN uses a multi-modal fusion for combining the three modalities' information.

One of the important aspects of TBN is the temporal binding strategy. Additionally, instead of combining the features from different modalities at a single point in time, it tries to investigate the interaction and evolution of these features over time. As a result, it considers data from different temporal offsets and builds a more comprehensive understanding of the action in the video.

Furthermore, the TBN uses a mid-level fusion approach, where each of the three modalities is processed by a separate convolutional layer at first. and before the classification, they combined to make a unified representation of the interactions of the features. In addition, it also takes advantage of sparse sampling by sampling features at specific time intervals during the videos, and by that, they try to do a trade-off between the temporal information and computation efficiency.

2.4.4 Context Base Egocentric Action Recognition

While most models in action recognition focus on the atomic processing of each action clip, without considering context the Multimodal Temporal Context Network (MTCN) [9] introduced an architecture that benefits the use of context for action recognition.

The MTCN architecture utilizes a window of size w to analyze sequences of

Figure 2.17: The temporal window sliding throughout the video, taken from the paper [9]

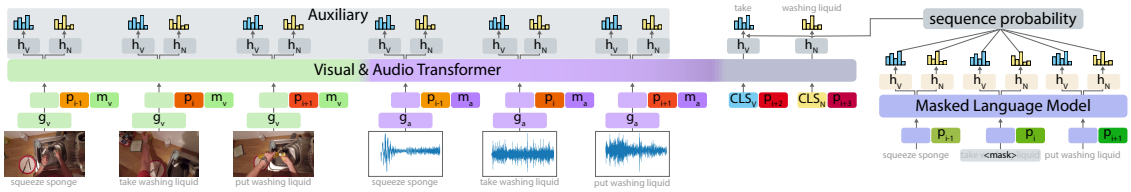


Figure 2.18: The MTCN processing the audio and visual information using transformers, taken from the paper [9]

actions within a video for single-action classification, as shown in Fig. 2.17. For each action, the MTCN model treats it as the central element of a window with size w. The model then processes the consecutive actions within this window to learn and exploit the temporal context, leading to better and more comprehensive results.

MTCN also focuses on the integration of the two modalities (RGB, and Audio), and to use both modalities efficiently it will introduce transformers encoders. These transformers encoder process the fusion of visual and audio features. In addition, both audio and visual information will be fed to transformers with positional embedding and using self-attention the transformers will provide a compact feature representation for both modalities. This process is illustrated in the Fig. 2.18.

Another important factor introduced by the MTCN was the incorporation of language models. This integration ensures that the model considers the broader context to improve its predictions. Specifically, the authors in [9] propose using a language model trained on verb and noun labels at test time. Additionally, they employ a top-k prediction strategy, where all possible action sequences are generated. Then, an exhaustive search is performed to identify the sequence with the highest score based on the language model's output. Finally, an impact factor is applied to weigh the language model's results with the visual prediction to determine the final model output.

While MTCN shows promising results in using context, their approach has several drawbacks, first of all, they do not take into account the domain shift, and the opportunity to use the sequence to mitigate the domain shift effects. Second, using the language model at the test time with brute force search is time-consuming and

not applicable to real-life scenarios.

2.5 Domain Generalization & Unsupervised Domain Adaptation

Egocentric action recognition models, though successful in their training data, often experience performance drops when deployed in new environments. This is because the visual characteristics of actions can differ significantly between training and deployment domains.

To address this challenge, researchers have developed techniques in domain adaptation (DA) [55, 56, 57] and domain generalization (DG) [58, 59, 60, 61, 62, 63] specifically for egocentric action recognition. This section will explore these models, starting with foundational adversarial unsupervised domain adaptation (UDA) methods (discussed in Section 2.5.1) and then progressing to DG methods.

2.5.1 General Adversarial Methods

The adversarial methods were initially introduced by the Domain-Adversarial Method [10]. The domain adversarial approach [10] was introduced to address challenges related to domain shifts in deep learning and computer vision. In this approach, as described in [10], a domain adaptation method is proposed that leverages unlabeled data from the target domain during training alongside source domain data. This unsupervised incorporation of target data aims to improve the model's robustness in the target domain.

Furthermore, inspired by the generative adversarial networks [64], the domain adversarial [10] introduced a domain classifier into the model, as illustrated in the Fig. 2.19. This domain classifier works in parallel to the main architecture to discriminate the target data from source data. However, the discriminator model will use the negative loss during backpropagation to trick the model into focusing on features that are more relevant to the task than features that are more relevant to the visual representation.

2.5.2 Video Adversarial Domain Adaptation

While the domain adversarial network [10] was a great model in domain adaptation, they were still a general model for general tasks in computer vision and deep learning. Therefore, new models introduced more specific domain adaptation models for tasks such as action recognition in video understanding. One of the most well-known models in this area is the Temporal Attentive Adversarial Adaptation Network (TA3N)[11].

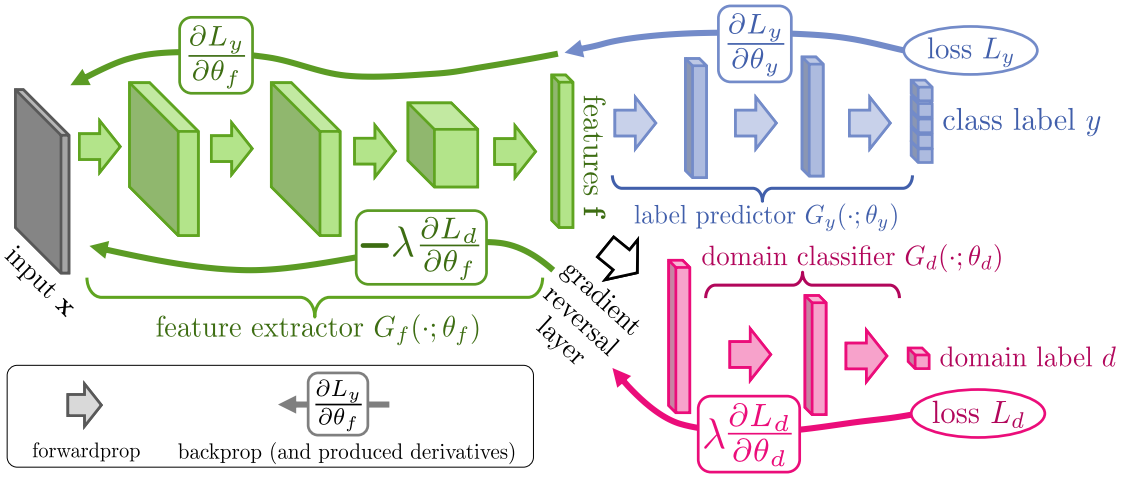


Figure 2.19: The architecture of domain adversarial model, taken from the [10].

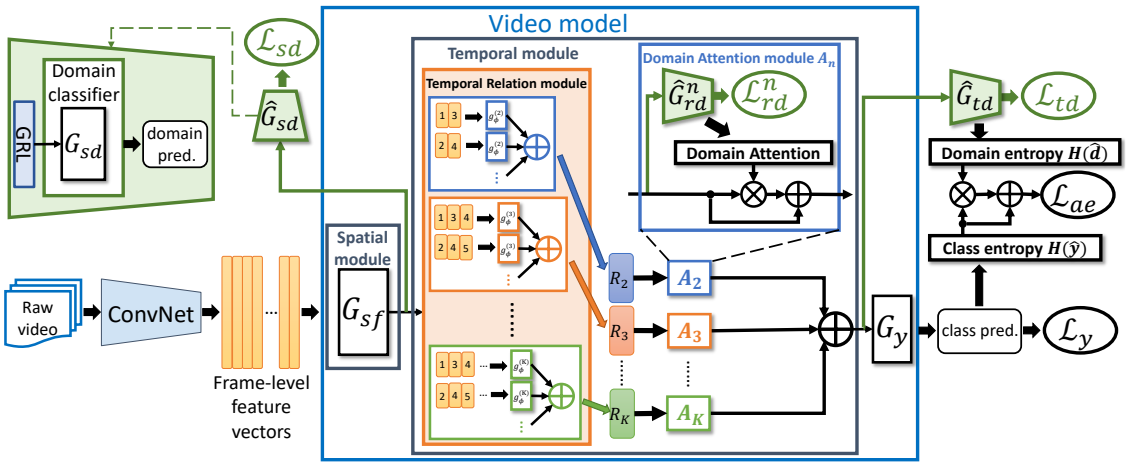


Figure 2.20: The TA3N architecture, taken from the paper [11].

The TA3N introduced a multi-layer adversarial mechanism to capture video information at different layers from frame level to feature layers. This mechanism allowed the TA3N to have a more comprehensive distinction between the two domains. In addition, TA3N use an auxiliary loss to combine the different adversarial layers' impact during training.

Looking at Fig. 2.20, we can see beside multiple layers of the discriminator for the reverse gradient. The TA3N also incorporates a temporal attention mechanism. This mechanism focuses on the videos' temporal dynamics as the essential and informative part. TA3N align the source and target data by using this attention mechanism. Therefore, the TA3N was able to present a more accurate video recognition when the model was applied to a new domain.

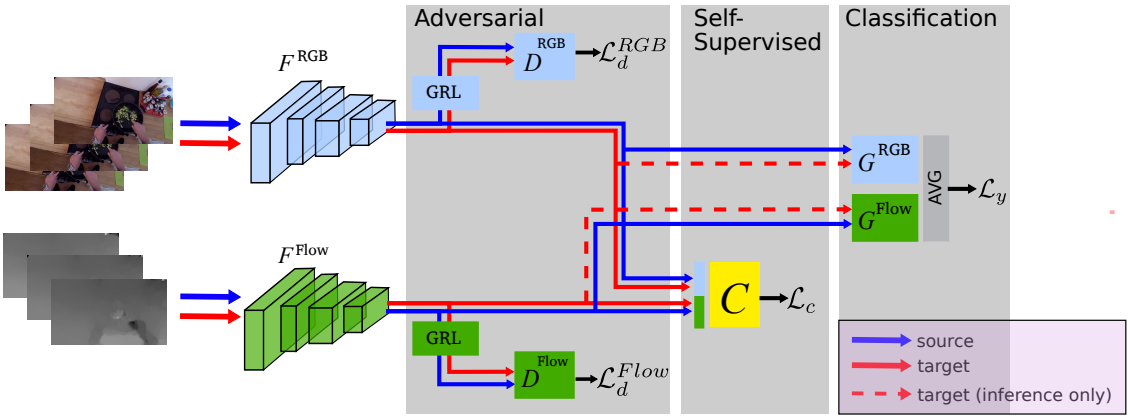


Figure 2.21: The MM-SADA architecture, taken from the paper [12]

2.5.3 Egocentric Domain Adaptation

TA3N showed promising results in video understanding, but the focus of this model was on third-person videos, without trying to mitigate challenges that exist in egocentric vision. Therefore, several models introduced for better approaching the domain shift problems in egocentric vision. The Multi-modal Self-Supervised Adversarial Domain Adaptation (MM-SADA) [12] was one of the models introduced for tackling the existing challenge in the domain gap between the training and testing data in egocentric vision.

Looking at Fig. 2.21, we can see this model introduced two modalities stream (RGB, and Optical Flow) for classification. Additionally, to better integrate these modalities it uses a self-supervised method, where it tries to map the optical flow of actions to their RGB instance representations. This task does not need any label and it will result in the model's inherent relationship between these modalities.

Furthermore, MM-SADA also tries to mitigate the domain shift effect using two adversarial sections for each modality. This approach's goal is to utilise shared feature extractor networks for both RGB and optical flow data. They focus on learning a generic representation for each modality and using the self-supervised focus of the training to align both modalities' representation into becoming more domain agnostic.

2.5.4 Adversarial free Domain Adaptation

In the world of UDA, some models are not based on adversarial methods, such as The Cross-modal Interactive Alignment (CIA) [13]. This model is another unsupervised domain adaptation model that introduced multi-modal learning, However, in contrast to previous works that focus on utilizing cross-modal by self-supervised learning, this model focuses on the cross-modal interaction first. In the [13], the writers by using cross-modal consensus, highlight the most transferable aspects of

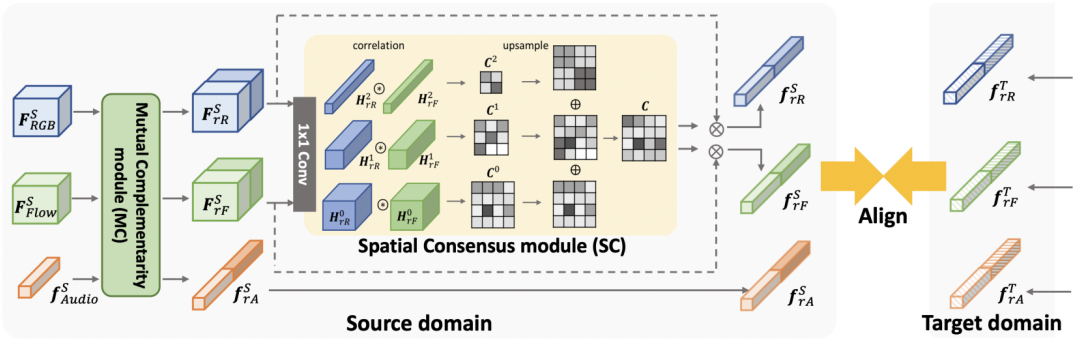


Figure 2.22: The presented CIA model in the paper of [13].

data.

In Fig. 2.22, the $\oplus$ denotes element-wise summation, $\otimes$ is element-wise multiplication, and $\circledast$ means the correlation operation.

Furthermore, considering Fig. 2.22, we can see the CIA model uses three modalities (RGB, Flow and Audio) at the same time. These modalities will be fed to the model as input, and by introducing the Mutual Complementarity module (MC) the CIA improve the transferability for every modality. This was achieved for each modality by learning the domain-transferable knowledge from different modalities. As a result, the CIA was able to leverage this cross-modal complementarity.

2.5.5 Domain Generalization

In Domain Generalization the target data is not available and the model only focuses on training using the source data. One of the DG well-known methods in egocentric action recognition is Relative Norm Alignment (RNA). The Relative Norm Alignment (RNA) [14] is another method that focuses on the egocentric action recognition robustness during the domain shift. In addition, the model focuses on the generalization of the egocentric models by introducing a new type of loss over different modalities to align them for a better prediction.

Furthermore, In the RNA method the author [14] introduces the problem of unbalanced contribution from the two modalities (RGB, and Audio) that they use for action classification. They indicate that during the training the model can become biased towards one of the modalities and lose useful information from the other one. As a result, they introduce the RNA loss to align the two modalities as follows:

$$\mathcal{L}_{RNA} = \left(\frac{E(h(X^V))}{E(h(X^A))} - 1\right)^2, \tag{2.17}$$

Where here the X^V and X^A are feature representations of visual and audio modalities, respectively. Also, the $E(h(X))$ is the mean value of L_2 feature norms of

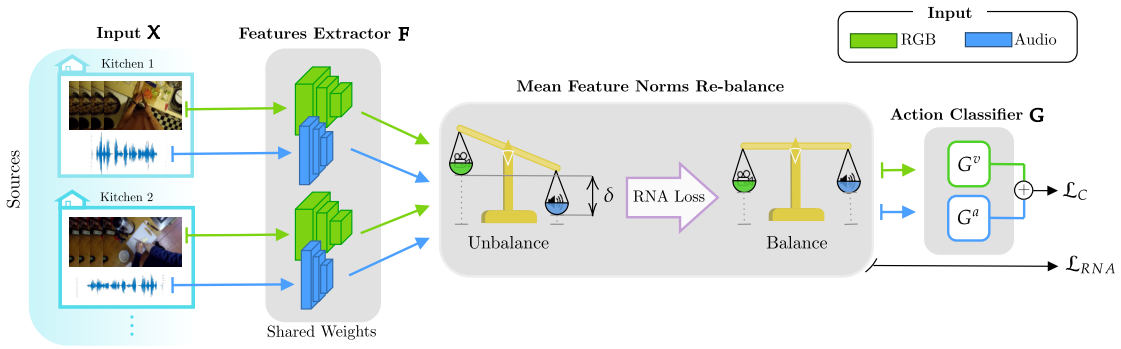


Figure 2.23: The presented architecture for RNA model, in the paper [14]

vector X.

Looking at Fig. 2.23, we can see how the author of the [14] illustrates the model structure during training. In addition, the model receives the two modalities as the input during the training time and the RNA loss will be calculated along the classification loss. Then the sum of both losses will be backpropagated throughout the model. This results in achieving a more balance presentation of the two modalities after finishing the training and during the test time.

Chapter 3

Sequential Domain Generalisation for Egocentric Action Recognition (SeqDG)

This chapter delves into our contribution to domain generalization in egocentric vision. We propose a novel approach, Sequential Domain Generalisation for Egocentric Action Recognition (SeqDG), that leverages context, specifically sequence information, to achieve domain-agnostic action classification and improve robustness against domain shift. We begin by discussing the motivation behind our method in Section 3.1. Next, Section 3.2 provides a general overview of the entire SeqDG method. Finally, subsequent sections introduce the details of each component.

3.1 Motivation

Domain shift poses a significant challenge for egocentric action recognition models. The challenges that exist in egocentric action recognition along with the lack of diversity in training data lead to a strong dependence on the specific environment where objects, interactions, and actors are captured. This heavy reliance reduces the model's ability to recognize the same actions in unseen visual settings. Furthermore, egocentric videos often involve human actors with distinct behaviours and interaction styles, adding another layer of complexity to the model's prediction task.

A key factor to consider when working on action recognition especially EAR is that human actions unfold sequentially, with each step contributing to a final goal. This highlights the importance of temporal context in action recognition.

For example, consider two people pouring milk in their separate kitchens (distinct environments). While their surroundings differ, they likely follow similar steps: opening the fridge, grabbing the milk container, removing the lid, and finally pouring the milk. Although specific details might vary based on factors like container shape or fridge door style, the overall sequence and completion of these actions remain consistent across environments, with minimal variations. As a result of this domain-agnostic behaviour, we tried to see how leveraging the temporal relationships between actions can enhance our network's ability to recognize actions even in diverse settings.

Furthermore, these action sequences hold the key to generalizability across visual domains. Because they are independent of the environment's layout or appearance, the model can focus on learning the core structure of action patterns. This ability to learn the gist of actions is crucial for recognizing them in new visual settings.

Additionally, analyzing sequences of continuous actions provides a richer understanding of user intent. A single action might be ambiguous on its own. By examining the sequence of actions, the model can gain a more comprehensive perspective on the user's ultimate goal.

As a result, we propose a domain generalisation method for egocentric action recognition (**SeqDG**) that benefits the similarity of action sequences across various visual environments to improve the generalisation of action recognition models. We introduced a visual-text sequence reconstruction objective called **SeqRec**, and we use the benefits of both text and visual modalities to extract the context for reconstructing the central action in the sequence. Knowing that textual information represents context with minimum effect from the environment, the textual narration is used to address the problem of domain shift in visual data. Moreover, we present a technique called **SeqMix** to increase the model robustness in changing the visual domain. This technique achieves its goal by mixing actions that have similar labels but they are different in terms of the visual domain.

3.2 Overview

This section presents an overall explanation of the proposed method (SeqDG), which aims to improve the robustness of egocentric action recognition models across different domains. The challenge addressed here is the performance drop these models experience when encountering unseen environments due to domain shift. Our approach leverages the similarity of action sequences with a similar goal across different visual environments.

Moreover, while using action sequences solely to classify a single action indicates promising results [9], it is still limited to overcoming the domain shift, due to being dependent on visual information to extract the sequence information. To address these limitations, we propose a visual-text sequence reconstruction objective (SeqRec, Section 3.6). SeqRec focuses on reconstructing the central action of

a sequence using contextual information from surrounding actions, incorporating both text and visual data. During training, SeqRec utilizes textual narrations to mitigate the domain shift that can occur in visual representations. Additionally, we propose a method for explicitly integrating information from multiple domains by mixing actions from different domains (SeqMix, Section 3.4). This process helps the model become more robust to changes in the visual appearance of actions. We refer to this overall approach as SeqDG. By combining these techniques, SeqDG encourages the model to learn visual representations for action recognition that are less dependent on specific domains.

3.3 Sequential Data Definition

In our work, each action a_i is defined by its video clip v_i, the free-form text narration t_i, the belonging visual domain d_i, and its pair labels *(verb, noun)*, that define the final label of action as follows $y_i = (y_i^V, y_i^N)$. Additionally, labels are from a set of N_v and N_n, verb and noun classes respectively, while the domain label d_i is one of K visual domains. Moreover, the actions $a_i = (v_i, t_i, y_i, d_i)$ also define the action sequence $\mathcal{S}_i$ to understand the action a_i by the ordered set of W actions centred around action a_i, represented by:

$$\mathcal{S}_i = \{a_{i-W/2}, \ldots, a_i, \ldots, a_{i+W/2}\}, \tag{3.1}$$

Furthermore, using pre-trained visual features and text extractors each action's visual frames and text narration in sequence will be converted to the corresponding visual appearance $\mathbf{X}_i^V \in \mathbb{R}^{W \times D_V}$ and textual $\mathbf{X}_i^T \in \mathbb{R}^{W \times D_T}$ features, where D_V and D_T correspond to the number of features for the video and the text respectively.

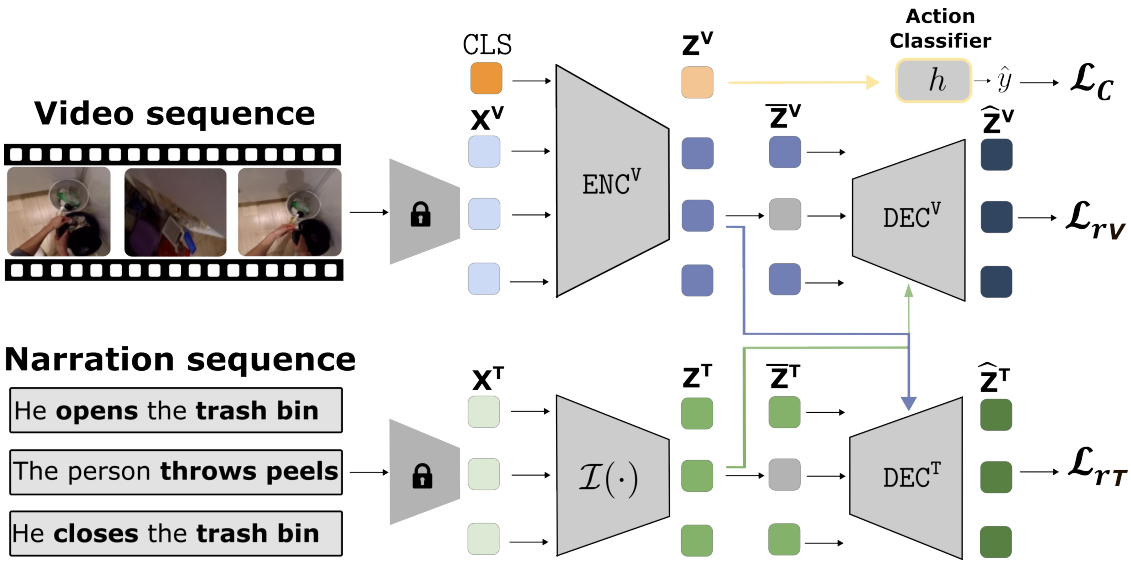


Figure 3.1: SeqDG architecture. We are given visual and textual inputs X^V and X^T. A classification token CLS is appended to the visual input for classification. Visual inputs are fed to an encoder $\texttt{ENC}^\texttt{V}$, resulting in intermediate visual embeddings Z_i^V, while textual features are passed through an identity function to get Z_i^T. The latter are masked ($\bar{Z}_i^V$ and $\bar{Z}_i^T$) and fed to two separate decoders $\texttt{DEC}^\texttt{V}$ and $\texttt{DEC}^\texttt{T}$ for visual and text reconstruction ($\mathcal{L}_{rV}$ and $\mathcal{L}_{rT}$). The transformed classification token $Z_{0:1}$ is fed to classifier h for action classification ($\mathcal{L}_C$).

3.4 SeqMix: Sequence Mix

Actions in sequences $\mathcal{S}_i$ are from the same source of videos, thus they borrow similar visual appearance from the environment. However, considering the context of the sequence, a domain-agnostic model should transform two similar sequences with the same context from different visual domains into a similar feature space. As a result, we introduced the Sequence Mix (SeqMix) to increase generalisation and decrease the domain dependence of the model. To do so we mix actions from different domains to stimulate the model to concentrate more on the semantics of the action itself and less on domain-specific visual biases. Given an action sequence $\mathcal{S}_i$, we replace action a_i (central action) in the sequence with a different action a_j from another sequence $\mathcal{S}_j$. The two actions are from different visual domains but share the same action label:

$$(a_i, a_j): \quad d_i \neq d_j \ \wedge \ y_i = y_j \tag{3.2}$$

here d_i, d_j and y_i, y_j are the domain and action labels of the two samples respectively.

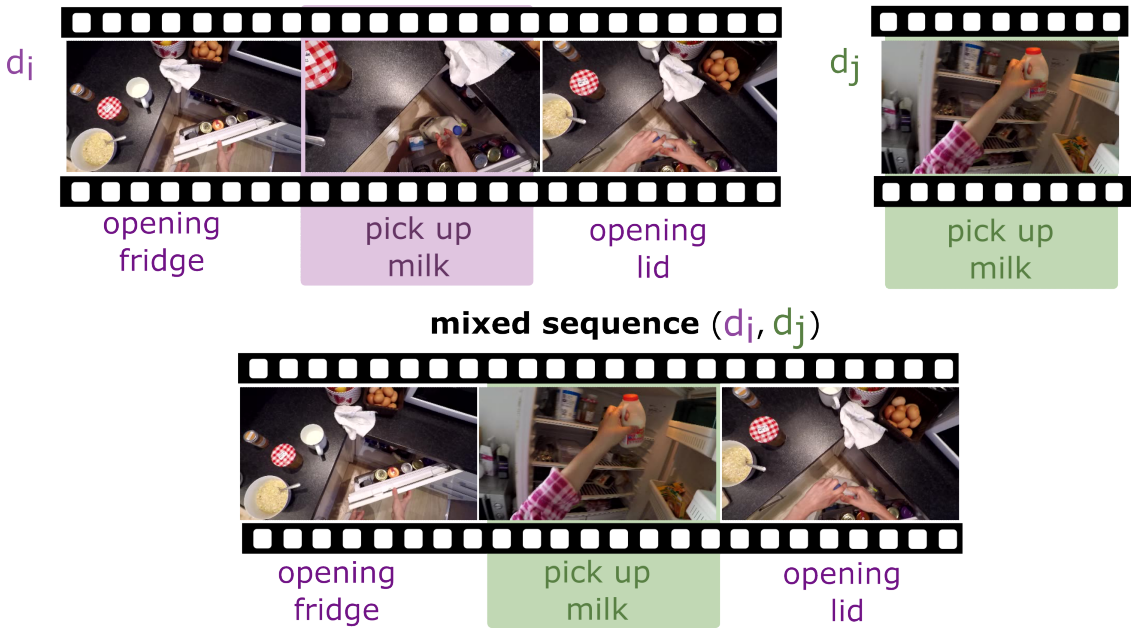


Figure 3.2: An example of SeqMix involves a sequence of actions from kitchen d_i (opening the fridge, picking up milk, opening the lid, pouring milk). We want to replace the "pick up milk" action to create a mixed sequence. Therefore, we choose an action video clip with the same label from Kitchen d_j and replace the equivalent action in the sequence from Kitchen d_i. This results in the final mixed sequence of kitchens d_i and d_j.

3.5 Sequence Feature Embedding

The model takes mixed sequences, denoted as S_i, as input. It utilizes these sequences to create a feature embedding space by considering the individual actions (S_i elements) within the sequence. To understand the input information more deeply, we leverage the attention mechanism from the Transformer architecture. Additionally, the visual and text features are fed into separate Transformer encoders, denoted as ENC^V and $\mathcal{I}$, respectively.

Since the Transformer's self-attention mechanism is insensitive to the order of elements, we employ positional encoding to retain information about the action sequence. A learnable positional embedding ($p \in \mathbb{R}^{W \times D_p}$) is applied to each action, tagging its absolute position within the sequence with respect to a temporal window of size W. Furthermore, two distinct learnable classification tokens (CLS_V and CLS_N, both in $\mathbb{R}^{D_A}$) are added to represent the predicted verb and noun of the central action. Finally, the input ($\mathbf{X}_i^V$ and $\mathbf{X}_i^T$) undergoes the following transformation:

$$\begin{aligned} Z_i^V &= \texttt{ENC}^{\texttt{V}}\left([X_i^V + p_i,\ CLS_V,\ CLS_N]\right) \\ Z_i^T &= \mathcal{I}\left(X_i^T\right) \end{aligned} \tag{3.3}$$

where the final outputs are $\mathbf{Z}_i^V \in \mathbb{R}^{(W+2)\times D}$ and $\mathbf{Z}_i^T \in \mathbb{R}^{W\times D}$, and they indicate the embeddings encoding that shows the relationships between different actions in the sequence.

We used the self-attention mechanism of transformers [36], to encourage the model to share information between the actions in the sequence. This allows the model to naturally learn relationships between different elements in a sequence. These relationships are built through a stack of multi-head attention, feed-forward (FF) layers, residual connection, and normalization layers (LN).

Let H^l be the features encoded at the l-th layer. Features produced at each layer of the transformer encoder are:

$$H^l_{attn} = \texttt{LN}(f_{SA}(H^l) + H^l) \tag{3.4}$$

$$H^{l+1} = \texttt{LN}(\texttt{FF}(H^l_{attn}) + H^l_{attn}) \tag{3.5}$$

where $f_{SA}(\cdot)$ is the self-attention operator defined as:

$$f_{SA} = \sigma\left(\frac{q(H^l)k(H^l)}{\sqrt{D}}\right)v(H^l) \tag{3.6}$$

where q, k and v are learnable projections of the input features, D is the size of the input features and $\sigma(\cdot)$ is the softmax function.

3.6 Sequence Reconstruction

We aim to enable the network to prioritize context when creating visual embeddings. To achieve this, we hypothesize that a domain-agnostic embedding can reconstruct a feature representation using only the surrounding context, due to the fact the representation of agnostic features is solely related to action, not visual representation. To test this hypothesis and encourage the network to become more domain-agnostic, we mask the visual and textual features of the central action (z_i^V and z_i^T) in a sequence. Subsequently, the model is tasked with reconstructing the masked action using decoder DEC^V (visual) and decoder DEC^T (Text) that employ cross-attention between the two decoders it reconstructs the masked feature embeddings. Moreover, the reconstruction process is guided by the unmasked features of the other modality. This means the masked visual features are reconstructed with the help of the surrounding textual features, and vice versa for textual features.

This process makes the network learn how to leverage the action context within the sequence. By strategically masking visual and textual features, the network is forced to rely on the surrounding information to reconstruct them. This fosters the development of domain-agnostic features, as the network learns to represent the action based on generic context rather than specific visual or textual cues.

To better understand the process of masking, by having the visual $\mathbf{Z}_i^V$ and textual $\mathbf{Z}_i^T$ features, we mask the central action of each feature embedding by putting it equal to zero:

$$\begin{aligned} \bar{\mathbf{Z}}_i^V &= \{z_{i-W/2}^V, \ldots, \bar{z}_i^V, \ldots, z_{i+W/2}^V\}, \\ \bar{\mathbf{Z}}_i^T &= \{z_{i-W/2}^T, \ldots, \bar{z}_i^T, \ldots, z_{i+W/2}^T\}, \end{aligned} \tag{3.7}$$

In the equation above, $\bar{z}_i^V$ and $\bar{z}_i^T$ represent the masked features of the central action in the sequence, for visual and textual feature embeddings respectively. Only the central action is masked during each iteration.

As previously mentioned, we employ two decoders: a visual feature decoder DEC^V and a textual feature decoder DEC^T. The visual feature decoder utilizes the combined information of the masked visual feature and the unmasked textual feature. Similarly, the textual feature decoder works with the masked textual feature and the unmasked visual feature. These decoders collaborate through a cross-attention mechanism.

$$\begin{aligned} \hat{\mathbf{Z}}_i^V &= DEC^V(\bar{\mathbf{Z}}_i^V, \mathbf{Z}_i^T) = softmax\left(\frac{q(\bar{\mathbf{Z}}_i^V)k(\mathbf{Z}_i^T)}{\sqrt{D}}\right) v(\bar{\mathbf{Z}}_i^V), \\ \hat{\mathbf{Z}}_i^T &= DEC^T(\bar{\mathbf{Z}}_i^T, \mathbf{Z}_i^V) = softmax\left(\frac{q(\bar{\mathbf{Z}}_i^T)k(\mathbf{Z}_i^V)}{\sqrt{D}}\right) v(\bar{\mathbf{Z}}_i^T), \end{aligned} \tag{3.8}$$

Here, q, k, and v represent learnable projections of the input features. $\hat{\mathbf{Z}}_i^V$ and $\hat{\mathbf{Z}}_i^T$ denote the reconstructed visual and textual features, respectively.

Due to the inherent differences between the two modalities (visual and textual), distinct loss functions are employed. For visual feature reconstruction, we utilize the L2 distance between the original and reconstructed features, denoted by $\mathcal{L}_{rv}$. For textual features, we employ cross-entropy loss between the action token represented by the reconstructed feature and the actual action, denoted by $\mathcal{L}_{rT}$. This approach encourages the visual feature encoder to generate features that capture contextual information based on the provided action sequence. Consequently, these encoded features enhance sequential representation, a critical factor for generalizing across diverse environments.

$$\begin{aligned} \mathcal{L}_{rv} &= MSE(\hat{\mathbf{Z}}_i^V, \mathbf{Z}_i^V), \\ \mathcal{L}_{rT} &= CrossEntropy(\hat{\mathbf{Z}}_i^T, \mathbf{Z}_i^T) \end{aligned} \tag{3.9}$$

3.7 Action Classification

In the final classification stage, we employ two separate classification heads, denoted as h^V and h^N, to classify the verb and noun, respectively, which are the key components of the action. These heads receive a classification summary of relevant information for classifying the central action and contextual information in the form of special tokens (CLS_V and CLS_N) from the action sequence obtained through the visual transformer. The CLS_V token represents the information needed to classify verb class, and CLS_N represents the information needed to classify noun class. Subsequently, the heads, h^V and h^N process these inputs and generate the verb and noun logits, $\hat{y}_i^V$ and $\hat{y}_i^N$, as the final outputs:

$$\hat{y}_i^V = h^V(CLS_V), \hat{y}_i^N = h^N(CLS_N) \tag{3.10}$$

Further, the two heads, h^V and h^N are trained using cross-entropy loss on the target action of classification, the central action a_i, classified for verbs ($\mathcal{L}_C^V$) and nouns ($\mathcal{L}_C^N$) separately. This is achieved by utilizing the corresponding ground truth action labels, which represent the pair of the verb and noun ($haty_i^V, \hat{y}_i^N$). Finally, the total classification loss is calculated as the mean of the verb and noun loss functions.

$$\begin{aligned} \mathcal{L}_C^V &= CrossEntropy(\hat{y}_i^V, y_i^V), \\ \mathcal{L}_C^N &= CrossEntropy(\hat{y}_i^N, y_i^N), \\ \mathcal{L}_C &= 0.5 \times (\mathcal{L}_C^V + \mathcal{L}_C^N), \end{aligned} \tag{3.11}$$

The network is trained by backpropagation of the sum of the classification and reconstruction losses, and the final objective is to minimize this overall loss:

$$\mathcal{L} = \mathcal{L}_C + \lambda_1 \mathcal{L}_{rV} + \lambda_2 \mathcal{L}_{rT} \tag{3.12}$$

where λ_1 and λ_2 weight the two reconstruction losses.

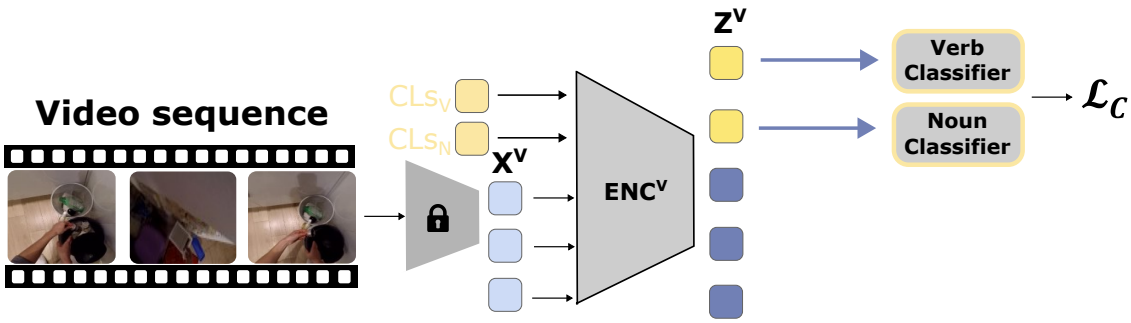


Figure 3.3: During the test time the model froze completely and only the visual classifier will classify the sequence to classify the middle action.

3.8 Inference Time

As illustrated in Fig. 3.3, the encoder-decoder architecture, trained with text narrations through the reconstruction task, guides the visual encoder, ENC^V, to more effectively integrate information from neighbouring actions. During testing, sequences comprised solely of video clips, v_i, are processed by ENC^V and the classifiers, h^V, and h^N. Predictions are made on the central action using a sliding window mechanism for sequence data of surrounding actions for testing. Importantly, textual annotations are not required at inference time, since the narrations are representations of labels which are not available. In addition, also by considering the goal of the training during the inference time we should have a robust visual encoder.

Chapter 4

Experiments

This chapter discusses the experiments conducted and the results obtained. Section 4.1, introduces the main benchmarks used for our evaluation. The following section 4.2, details the general experimental setup and implementation specifics. We then present the results in sections 4.4 and 4.3, covering both intra-domain and cross-domain settings. Finally, section 4.5 analyzes the effectiveness of each component.

4.1 Datasets

In order to show the effectiveness of our approach, we used three datasets, which indicate the model performance over these datasets. These datasets are considered some of the state-of-the-art benchmarks in egocentric action recognition, and the diversity of their data in different domains provides us with a good assessment of our model performance. The datasets we used are the Epic-Kitchen-55 [65], Epic-Kitchens-100 [1], and Extended Georgia Tech Egocentric Activity (EGTEA) [2]. Each dataset will be discussed in detail in the following sections.

4.1.1 Epic-Kitchens-55

Epic-Kitchens-55 [65] is a well-known dataset in egocentric vision. This dataset consists of 32 environments and offers a great opportunity for assessing the domain shift problem. A well-known split from this dataset, commonly used to analyze approaches in domain generalization and adaptation, is the MM-SADA split [12]. This split provides data in three distinct domains over eight different verbs as the action labels for this dataset ('put', 'take', 'open', 'close', 'wash', 'cut', 'mix', and 'pour'). These three domains are referred to as D1, D2, and D3, as illustrated in Fig. 4.1 with samples from each domain. Furthermore, the eight class labels are the 80% of training action segment for these domains, This approach by [12]

Figure 4.1: The three domains of visual data for Epic-Kitchens-55 presented in the MM-SADA [12] paper.

was to ensure adequate samples per domain and class, without balancing them in the training set. Additionally, Fig 4.3 indicates the eight classes throughout the domains are imbalanced to offer additional challenges to the problem.

4.1.2 Epic-Kitchens-100

Epic-Kitchens-100 [1] is another well-known egocentric vision dataset, This dataset is an extended version of the Epic-Kitchens-55 with more than 100 hours of collected videos, with 20M frames, and 90k actions by pairing 300 nouns and 97 verbs in 700 variable-length videos, that had been captured in 45 different environments. This dataset has a more complete annotation of fine-grained actions in comparison to Epic-Kitchens-55 with +128% more action segments.

The diversity of this dataset introduces challenges, particularly in unsupervised domain adaptation (UDA), and domain generalization. To address this, a special split of the dataset was introduced, dividing the data into two sets: source and target. Each set consists of multiple domains but remains distinct from the other set. Notably, the source set provides labelled actions for both training and validation, while the target set lacks labels for training, and target validation is only used to assess the strength of approaches for adaptation and generalization.

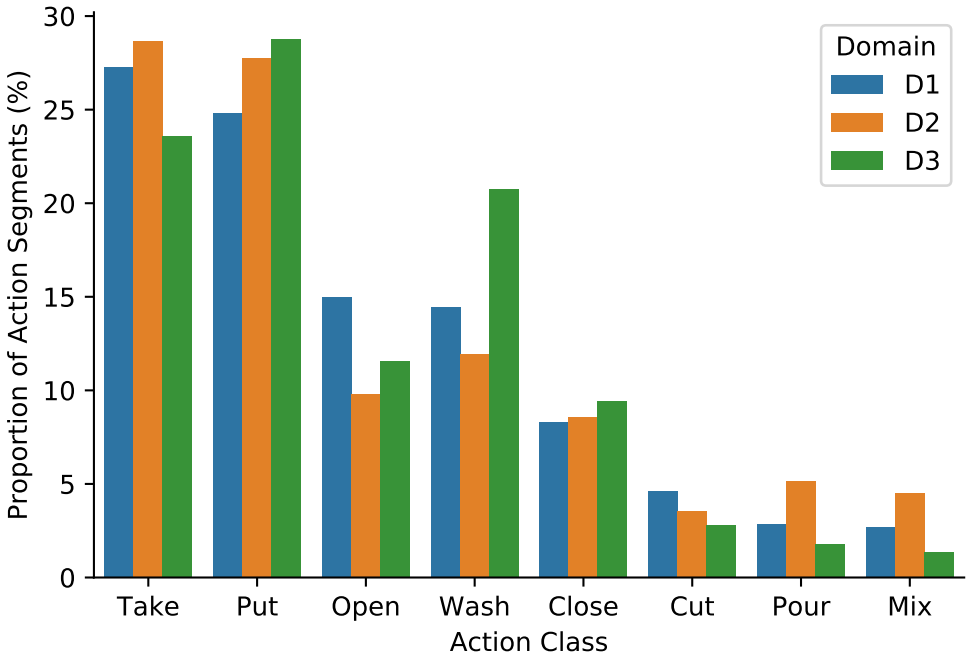


Figure 4.2: Distribution of actions in Epic-Kitchens-55 for the MM-SADA [12] split.

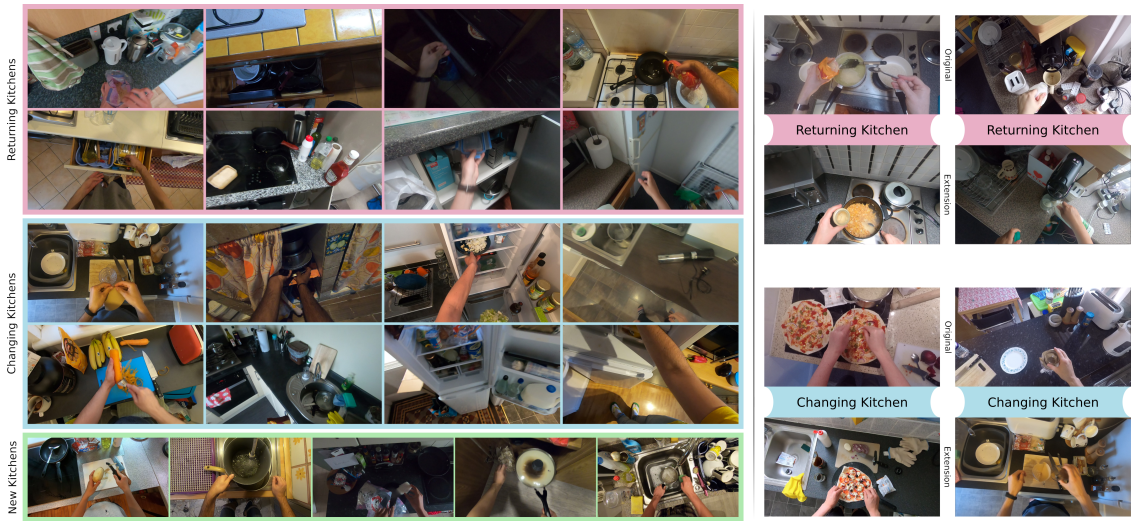


Figure 4.3: Samples of new data of Epic-Kitchens-100, where new kitchen, some old kitchen re-captured data or changed.

Figure 4.4: The EGTEA[2] dataset sample frames.

4.1.3 Extended Georgia Tech Egocentric Activity (EGTEA)

The final well-known benchmark we used to assess the power of our approach, not for domain generalization but for intra-domain performance, is the Extended Georgia Tech Egocentric Activity (EGTEA) dataset [2]. EGTEA is a large-scale egocentric action recognition dataset featuring high-definition (HD) videos (1280x960). It encompasses 28 hours (de-identified) of untrimmed videos with approximately 10,000 annotations of actions from a set of 106 action labels related to cooking activities. These annotations come from 86 unique sessions performed by 32 subjects

4.2 Implementation details

In this section, we will discuss the implementation setting and the environments where experiments are done for different benchmarks. Moreover, we will discuss the data input dimensions, training details, and pre-trained model that had been

used in our work.

In the case of Epic-Kitchen-55 (MM-SADA split), we use the TSM [7] features pre-trained on each of the three domains and extracted for other domains. The features are extracted into 5 clips with a dimension of 2048. Moreover, we used all 5 clips to train our model.

Furthermore, For Epic-Kitchen-100 the TBN [8] features are pre-trained on the source split of the dataset. The extracted features contain 25 clips with a dimension of 1024. However, to train our method we only used 5 clips that were uniformly sampled from the 25 clips.

For EGTEA features the SlowFast [6] pre-trained features released by the authors of MTCN [9]. Additionally, the 10 clips for each action are extracted, and each clip has a dimension of 2304. We used all 10 clips for the EGTEA training dataset.

During the training of our approach, to capture the gist information of all clips we used a Temporal Relation Network (TRN) [19] layer for each action. To encode the narrations for the textual part of training the BERT pre-trained model is used as described in [37]. Using the BERT gives us the textual feature size of 768, and to unify the dimension of the textual and visual features we project the latter using a fully connected layer to the same dimension as textual (768-D).

Finally, All models are trained with the SGD optimiser, using batch size 32 and an initial learning rate of 0.005. Training lasts 100 for Epic-Kitchen-55 and Epic-Kitchen-100, and 50 epochs for EGTEA. The learning rate is decreased by a factor of 10 at epochs 50 and 75 for Epic-Kitchen-55 and Epic-Kitchen-100, and at epochs 25 and 38 for EGTEA.

4.3 SeqDG Cross-domain Results

This section is dedicated to exploring how our method improves action recognition in the face of domain shift. The method's domain generalization capabilities will be assessed against other well-known methods in domain adaptation and domain generalization. To accomplish this, we will use Epic-Kitchen-100 and Epic-Kitchen-55 datasets.

In Epic-Kitchen-100, we will assess the improvement of our model performance compared to a baseline, which is a simple classifier on top of the inputs. Then, we will compare our gain on action recognition to the gain achieved by other methods on action recognition based on their reported baselines in their respective papers. At the same time, for Epic-Kitchen-55, we will only assess the model on the three domains (D1, D2, and D3) and indicate the improvement according to the baseline of each domain.

4.3.1 Comparison with state-of-the-art

In Table 4.1 (Epic-Kitchen-100), our model achieves superior results with approximately 2.4% higher accuracy on top-1 action classification. This indicates its ability to handle domain shift likely due to its more domain-agnostic nature. For a fair comparison, we employed three modalities (RGB, Audio, and Flow) based on prior findings in other research. Additionally, to ensure a direct comparison with MTCN [9], we report results using only RGB and Audio modalities, achieving a roughly 2.1% improvement over the baseline.

Furthermore, in Table 4.2, we can see our model achieves about 47.6% on average over all the domains. In addition, compared to the Baseline we achieved about 4.4% improvement on the mean performance. These results can indicate our model's ability in domain-agnostic prediction during domain shift in this dataset.

Method	Sequence	Modalities			Top-1 Accuracy (%)		
		RGB	Flow	Audio	Verb	Noun	Action
UDA							
Source Only	-	✓	✓	✓	46.7	27.8	19.2
TA3N [1]	-	✓	✓	✓	48.5	28.9	19.6 (▲ +0.4)
Source Only	-	✓	✓	✓	47.1	27.4	19.0
MM-SADA [12]	-	✓	✓	✓	48.4	28.3	19.3 (▲ +0.3)
Source Only	-	✓	✓	✓	47.6	28.4	19.6
CIA [13]	-	✓	✓	✓	48.3	29.5	20.3 (▲ +0.7)
Source Only	-	✓	✓	✓	46.7	26.7	18.2
RNA [20]	-	✓	✓	✓	**50.8**	29.1	20.0 (▲ +1.8)
DG							
Source Only	-	✓	✓	✓	47.2	27.4	19.0
MM-SADA (SS) [12]	-	✓	✓	✓	47.8	27.9	19.2 (▲ +0.2)
Source Only	-	✓	✓	✓	46.7	26.7	18.2
RNA [20]	-	✓	✓	✓	50.7	27.9	19.8 (▲ +1.6)
Source Only	✓	✓	✗	✓	38.3	22.4	14.3
MTCN [9]	✓	✓	✗	✓	39.8	25.5	14.8 (▲ +0.5)
Source Only	✓	✓	✗	✓	38.7	23.8	14.8
SeqDG	✓	✓	✗	✓	41.3	26.8	16.9 (▲ +2.1)
Source Only	✓	✓	✓	✓	46.4	26.6	18.2
SeqDG (No text)	✓	✓	✓	✓	47.5	28.9	19.5 (▲ +1.3)
SeqDG	✓	✓	✓	✓	49.1	**29.8**	**20.6** (▲ +2.4)

Table 4.1: Comparison with state-of-the-art in the Cross-Domain setting of the Epic-Kitchen-100 UDA benchmark (target validation split). Models are evaluated in terms of Top-1 and Top-5 Verb, Noun and Action accuracy (%).

Setting	$D1 \rightarrow D2$	$D1 \rightarrow D3$	$D2 \rightarrow D1$	$D2 \rightarrow D3$	$D3 \rightarrow D1$	$D3 \rightarrow D2$
Baseline	43.3	42.0	42.5	43.4	43.2	45.0
SeqDG	45.3(▲ +2.0)	47.3(▲ +5.3)	48.5(▲ +6.0)	46.9(▲ +3.5)	46.2(▲ +3.0)	51.4(▲ +6.4)

Table 4.2: Epic-Kitchen-55 Results

4.4 SeqDG Intra-domain Results

This section explores how our method improves intra-domain action recognition. The method's domain generalization capabilities not only enhance performance during domain shifts but also lead to more robust predictions within the same domain. This is because considering context remains valuable during testing within the same domain due to the increased similarity in how actions are performed by the same actor.

Moreover, to benchmark our performance, we present the results of a baseline model. This baseline uses a simple classifier head on top of the features, without any of the core components of our proposed method. On the EPIC-Kitchens-100 Dataset, our approach achieved a higher top-1 action accuracy. Our method, in comparison, captured an average improvement of +5.2 on top-1 accuracy.

Finally, to better evaluate our model's performance against state-of-the-art methods, we compare our method on both the Epic-Kitchen-100 and EGTEA datasets. We include the MTCN model [9], which integrates sequence and context information. Additionally, we compare our method with well-known egocentric action recognition methods on the EGTEA dataset.

4.4.1 Comparison With State-Of-The-Art

In Table 4.3, we indicate the results to compare our method not only by the baseline but also in comparison with MTCN [9], and also the † is used to show LM as test time for fair comparison with MTCN. Additionally, It had been indicated, as our method was able to achieve +5.2 according to the baseline, and in comparison to MTCN [9] which was able to achieve +3.5 in intra-domain, we were able to outperform this method.

Further, in Table 4.4 we showed the model's power to outperform other state-of-the-art. We achieved 74.1% accuracy on the Top-1 accuracy of action. Moreover, the performance of our model without a text-based test-time language model (indicated by †) is about 1.5% better than the best performance achieved by other methods, specifically MTCN [9]. When considering the test-time language model, we achieved a further improvement of 0.6% compared to MTCN [9].

Method	Sequence	Modalities			Top-1 Acc. (%)			Δ Mean Acc. (%)
		RGB	Flow	Audio	Verb	Noun	Action	
Baseline	✓	✓	✗	✓	60.0	42.8	30.8	-
MTCN [9]	✓	✓	✗	✓	60.6	48.2	33.0	+2.2
MTCN† [9]	✓	✓	✗	✓	61.5	49.3	34.3	+3.5
Baseline	✓	✓	✗	✓	60.1	42.8	31.1	-
SeqDG	✓	✓	✗	✓	63.9	49.1	36.1	+5.0
SeqDG†	✓	✓	✗	✓	63.6	49.7	36.3	+5.2

Table 4.3: EPIC-Kitchens-100 Intra-domain Comparison

Method	Top-1 Acc. (%)	Mean Class Acc. (%)
Kapidis et al. [53]	68.9	60.5
Lu et al. [66]	68.6	60.5
SlowFast [6]	70.4	61.9
Min et al. [67]	68.5	62.8
MTCN [9]	72.5	64.8
SeqDG	74.0	66.5
MTCN† [9]	73.5	65.8
SeqDG†	**74.1**	**66.9**

Table 4.4: EGTEA state-of-the-art comparison

4.5 Ablations

In this section, we will present an ablation study of the model components, analyzing the effect of each part on overall performance. We will assess the results of each component in subsection 4.5.1. Then, in the next subsection 4.5.3, we will analyze the impact of different sequence lengths. Following that, we will discuss the reasoning behind hyper-parameter variation. Subsequently, in subsection 4.5.6, we will present the results of using different modalities. Finally, the next two subsections, 4.5.7 and 4.5.8, will detail the results of different language models and qualitative results, respectively.

4.5.1 SeqDG components

In Table 4.5, we present the results of our model using only RGB input to isolate the effects of each component. We begin with a baseline performance on RGB-only data. The table then shows how the addition of each component incrementally improves the model's performance.

Looking at Table 4.5, we can see that the baseline model, which does not include a sequence, performs the worst. Adding a sequence to the model results in a 0.4%

Sequence	SeqMix	$\mathcal{L}_{rA}$	$\mathcal{L}_{rT}$	Verb	Noun	Action
-	-	-	-	33.5	21.6	11.6
✓	-	-	-	32.9	22.7	12.0
✓	✓	-	-	33.9	23.2	12.1
✓	✓	✓	-	33.5	23.7	12.4
✓	✓	-	✓	34.2	23.6	12.2
✓	✓	✓	✓	**34.3**	**24.2**	**12.8**

Table 4.5: Ablation study on the different components of SeqDG on EK-100 in terms Top-1 accuracy (%) using RGB information only.

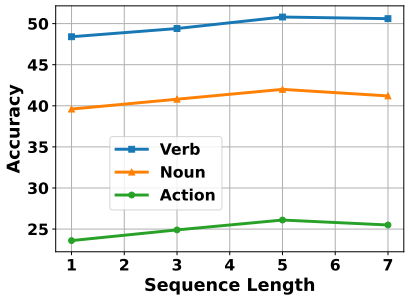
improvement over the baseline. Further improvement is achieved by incorporating SeqMix, yielding a total of 0.5% improvement relative to the baseline. Finally, adding visual reconstruction loss and text reconstruction loss leads to additional improvements of 0.8% and 0.6%, respectively, compared to the baseline model. The most significant improvement, however, is achieved by combining all components, resulting in a 1.2% gain over the baseline.

4.5.2 Epic-Kitchen-100 UDA Challenge

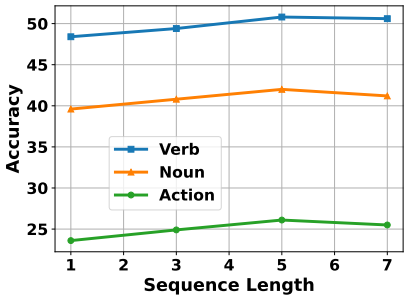
To further demonstrate our work's effectiveness, particularly the sequence mix methodology, we participated in the EPIC-KITCHENS-100 Unsupervised Domain Adaptation (UDA) Challenge for Action Recognition at the 2023 Conference on Computer Vision and Pattern Recognition (CVPR). We achieved second place in this specific task and ranked fourth overall (as shown in Table 4.6). Notably, we achieved these results using only a sequence-based prediction approach combined with our SeqMix method. This success highlights the strength of our approach when employing multiple modalities with different backbones.

Rank	Method	Verb	Noun	Action
1	Ns-LLM	**58.22**	40.33	**30.14**
2	VI-I2R [68]	57.89	40.07	30.12
3	Audio-Adaptive-CVPR2022 [69]	52.95	**42.26**	28.06
4	**sshayan [70]**	58.11	35.89	27.72
5	plnet [71]	55.51	35.86	25.25

Table 4.6: Top-1 accuracy on UDA's EPIC-Kitchens-100 leaderboard. Our submission is highlighted.



(a) Intra-domain sequence length effect.



(b) Cross-domain sequence length effect.

4.5.3 Sequence length

Figure 4.5b, and 4.5a illustrate the relationship between performance and the number of actions in a sequence, also known as sequence length. The best results are achieved with sequences containing five actions. Performance decreases for both shorter and longer sequences. This can be explained by the limitations of shorter sequences. They might not provide enough temporal context, which refers to the order and timing of the actions. Conversely, longer sequences might not be universally applicable across different domains. Additionally, they might introduce the dimensionality curse by bringing irrelevant actions that are not related to the core functionality of the sequence and cannot be effectively understood by simply looking at the other actions in the sequence. Due to these factors, we have chosen a sequence length of five for all our experiments.

4.5.4 Sequences Similarity across the domain

To test our hypothesis, we also calculated the number of distinct sequences repeated across the two domains in Epic-Kitchen-100 [1]. Fig. 4.6 shows the number of sequences repeated between different domains in the UDA split of Epic-Kitchen-100 [1]. We calculated the sequence number based on the sequence of verbs, nouns, and actions separately.

Furthermore, we extended this analysis to consider not only the full sequences but also all repeated subsequences, down to a length of two. As evident in Fig. 4.6, the number of action sequences and noun sequences shows minimal improvement beyond a length of five. This finding supports our choice of a five-action window for sequence modelling based on the results of our hyperparameter testing.

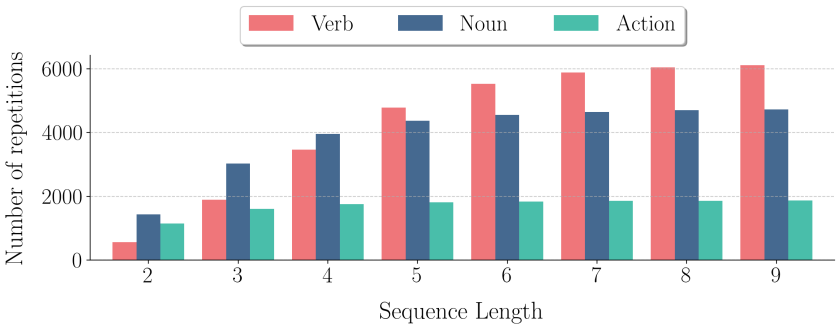


Figure 4.6: The number of repeated sequences with different numbers of actions between two different domains in the UDA split of Epic-Kitchen-100[1].

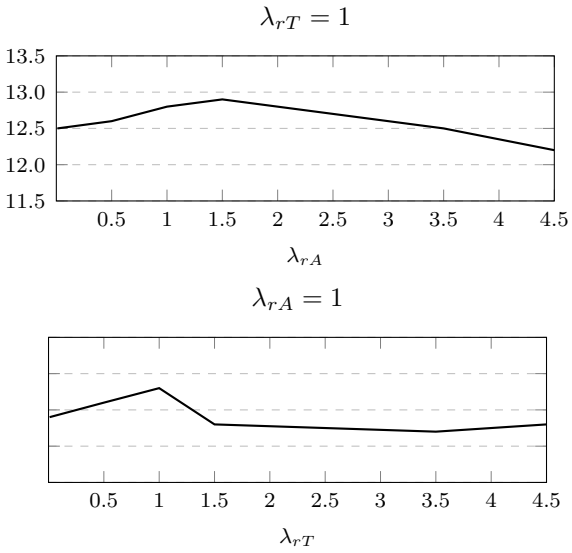


Figure 4.7: Parameter analysis of the weights associated with the visual and textual reconstruction losses of SeqDG (RGB).

4.5.5 Hyper-parameters variation

Figure 4.7 explores the impact of varying the weights assigned to the visual and textual reconstruction losses within a multimodal learning framework. The experiment involves systematically increasing the weight of one loss function while keeping the other fixed. This ablation study allows us to isolate the influence of each loss term on the overall performance of the model. The results demonstrate that the model achieves its optimal performance when the reconstruction objective prioritizes both modalities equally, assigning a weight of 1.0 to both the visual and textual reconstruction losses. This suggests that the model benefits from learning a robust joint representation that captures the complementary information present in both visual and textual data.

Setting	RGB	Flow	Audio	Verb	Noun	Action
Source Only	✓	-	-	33.5	21.6	11.6
SeqDG	✓	-	-	34.3	24.2	12.8
Source Only	✓	✓	-	42.1	24.3	15.3
SeqDG	✓	✓	-	45.0	28.4	18.0
Source Only	✓	-	✓	38.7	23.8	14.8
SeqDG	✓	-	✓	41.3	26.8	16.9
Source Only	✓	✓	✓	46.4	26.6	18.2
SeqDG	✓	✓	✓	**49.1**	**29.8**	**20.6**

Table 4.7: Comparison of different modalities on EK-100.

4.5.6 Modalities Ablation

Beyond visual appearance, other modalities inputs like audio and optical flow can also be susceptible to domain shift in various ways. For instance, the way objects interact with their environment, as captured by optical flow, and the sounds they produce, as captured by audio, can be influenced by their shape and material composition.

Table 4.7 showcases the performance of SeqDG on multimodal tasks, where we observe even more significant improvements compared to using a single modality alone. This finding not only suggests that our method can be effortlessly applied to other sensory inputs but also highlights the value of the complementary information provided by these additional modalities in aiding the reconstruction process.

4.5.7 Language Model

To showcase the robustness of SeqDG to variations in textual encoders, we evaluate its performance using different language models for extracting the textual features employed during reconstruction. Even though these language models have vastly different sizes, we observe similar accuracy for verb and noun recognition. This reinforces the notion that text in SeqDG serves primarily as a guiding force to mitigate domain shift during reconstruction, rather than being a core component of the final inference process. The reasults is indicated in

Model	Verb Acc. (%)	Noun Acc. (%)	Action Acc. (%)
Source Only	33.5	21.6	11.6
CLIP [72]	34.4	23.2	12.3
MiniLM [73]	**34.8**	23.6	12.4
BERT [37]	34.3	**24.2**	**12.8**

Table 4.8: Comparison of different Language Models for SeqDG.

4.5.8 Qualitative Results

Figure 4.8: Qualitative examples showing success and failure cases of SeqDG.

To illustrate the power of leveraging temporal context, Figure 4.8 showcases qualitative examples where a baseline model, lacking the ability to analyze sequences, would falter. In contrast, our approach successfully predicts the correct label by capitalizing on the informative sequence of actions. For instance, a baseline model might struggle to identify the final action of "making a sandwich" if presented solely with an image of spreading mayonnaise on bread. However, by analyzing the entire sequence – retrieving bread, applying condiments, adding fillings – our model can accurately predict the final goal.

Additionally, it's important to acknowledge limitations. While sequence analysis empowers our approach, there can be scenarios where the central actions within a sequence hold a weak connection to the surrounding actions. Imagine a scenario where someone retrieves a set of objects from a cupboard. Here, the act of retrieving objects itself doesn't provide much context about the specific purpose. In such cases, our approach might extract limited meaningful information from the sequence, potentially leading to an incorrect prediction. This highlights the importance of considering both the strengths and limitations of sequence-based models when designing robust AI systems.

Chapter 5

Conclusions

In this work, we tackled the challenge of how domain shift affects egocentric action recognition. Egocentric videos, by their nature, are challenging data. As a result, they tend to have a strong dependency on the visual information of the environment. Recognizing that these dependencies can reduce the performance when the environment changes, we proposed a solution to mitigate them. This solution leverages context as a moderator of visual representation.

We argued that context acts as a robust element because it's less affected by shifts in the model environment. While an object's interaction with an actor can change based on the environment (think frying an egg on a camping stove vs. a home kitchen), the underlying reasoning behind the action often remains the same. To illustrate, consider cooking fried eggs. The sequence of actions needed – picking up the egg, cracking it, pouring it into a pan – remains consistent for the same goal, with a lower probability of changing during an environment shift. Therefore, inspired by this reasoning, we introduced SeqDG, which increases robustness to domain shift.

Similar to unsupervised domain adaptation (UDA), SeqDG tackles domain generalization (DG). However, unlike UDA, SeqDG assumes that data from the unseen domain is completely unavailable. To address this challenge, we leveraged the training data to become a robust model by taking the best advantage of available data, and SeqDG demonstrates this improved robustness through classification using sequences rather than individual actions. This approach highlights the context's role in boosting domain generalization. We introduce the novel mixed sequence method (SeqMix), which constructs sequences by combining actions from different domains while maintaining the label order from the seen training data. This method compels the model to learn domain-agnostic features. Additionally, we employ visual-textual reconstruction (SeqRec) to investigate the influence of the language model during training and its contribution to building a more robust model for domain generalization.

Finally, by presenting our results on different benchmarks (Epic-Kitchen-100 [1],

and EGTEA [2]), we showed the superiority of our model by achieving +2.4% accuracy on top-1 action in Epic-Kitchen-100. By analyzing the results, we are also able to indicate the effect of the sequence along which each element is presented in the SeqDG. This finding highlights the potential of context in DG, suggesting that future research could explore how to leverage context for improved model performance across the different domains.

Bibliography

[1] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, E. Kazakos, J. Ma, D. Moltisanti, J. Munro, T. Perrett, W. Price, *et al.*, "Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100," *International Journal of Computer Vision*, pp. 1–23, 2022.

[2] Y. Li, M. Liu, and J. M. Rehg, "In the eye of beholder: Joint learning of gaze and actions in first person video," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 619–635, 2018.

[3] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, C. Fuegen, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. M. Farinella, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik, "Ego4d: Around the World in 3,000 Hours of Egocentric Video," in *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] R. Pramoditha, "The concept of artificial neurons (perceptrons) in neural networks," 2021.

[5] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[6] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

[7] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF international conference on*

computer vision, pp. 7083–7093, 2019.

[8] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Epic-fusion: Audio-visual temporal binding for egocentric action recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5492–5501, 2019.

[9] E. Kazakos, J. Huh, A. Nagrani, A. Zisserman, and D. Damen, "With a little help from my temporal context: Multimodal egocentric action recognition," *arXiv preprint arXiv:2111.01024*, 2021.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016.

[11] M.-H. Chen, Z. Kira, G. AlRegib, J. Yoo, R. Chen, and J. Zheng, "Temporal attentive alignment for large-scale video domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6321–6330, 2019.

[12] J. Munro and D. Damen, "Multi-modal domain adaptation for fine-grained action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 122–132, 2020.

[13] L. Yang, Y. Huang, Y. Sugano, and Y. Sato, "Interact before align: Leveraging cross-modal knowledge for domain adaptive action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14722–14732, 2022.

[14] M. Planamente, C. Plizzari, E. Alberti, and B. Caputo, "Domain generalization through audio-visual relative norm alignment in first person action recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1807–1818, 2022.

[15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[16] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick, "Long-term feature banks for detailed video understanding," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 284–293, 2019.

[17] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *Advances in neural information processing systems*, vol. 27, 2014.

[18] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[19] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 803–818, 2018.

[20] M. Planamente, C. Plizzari, S. A. Peirone, B. Caputo, and A. Bottino, "Relative norm alignment for tackling domain shift in deep multi-modal classification," *International Journal of Computer Vision*, pp. 1–21, 2024.
[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[22] T. M. Mitchell, "Machine learning," 1997.
[23] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
[24] F. Rosenblatt *et al.*, *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, vol. 55. Spartan books Washington, DC, 1962.
[25] M. Minsky and S. A. Papert, *Perceptrons, reissue of the 1988 expanded edition with a new foreword by Léon Bottou: an introduction to computational geometry*. MIT press, 2017.
[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.
[27] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
[28] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 20, no. 2, pp. 215–232, 1958.
[29] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.
[30] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, pp. 462–466, 1952.
[31] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Transactions on Electronic Computers*, no. 3, pp. 299–307, 1967.
[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
[35] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *arXiv preprint arXiv:1912.05911*, 2019.
[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of

deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[38] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[39] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[40] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[41] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[42] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[43] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, *et al.*, "Lamda: Language models for dialog applications," *arXiv preprint arXiv:2201.08239*, 2022.

[44] S. Sudhakaran and O. Lanz, "Convolutional long short-term memory networks for recognizing first person interactions," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2339–2346, 2017.

[45] S. Sudhakaran and O. Lanz, "Attention is all we need: Nailing down object-centric attention for egocentric activity recognition," *arXiv preprint arXiv:1807.11794*, 2018.

[46] S. Sudhakaran, S. Escalera, and O. Lanz, "Lsta: Long short-term attention for egocentric action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9954–9963, 2019.

[47] A. Furnari and G. M. Farinella, "Rolling-unrolling lstms for action anticipation from first-person video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4021–4036, 2020.

[48] M. Planamente, A. Bottino, and B. Caputo, "Self-supervised joint encoding of motion and appearance for first person action recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 8751–8758, IEEE, 2021.

[49] M. Ma, H. Fan, and K. M. Kitani, "Going deeper into first-person activity recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1894–1903, 2016.

[50] S. Sudhakaran, S. Escalera, and O. Lanz, "Gate-shift networks for video action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1102–1111, 2020.

[51] A. Cartas, J. Luque, P. Radeva, C. Segura, and M. Dimiccoli, "Seeing and hearing egocentric actions: How much can we learn?," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[52] S. Singh, C. Arora, and C. Jawahar, "First person action recognition using deep learned descriptors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2620–2628, 2016.

[53] G. Kapidis, R. Poppe, E. van Dam, L. Noldus, and R. Veltkamp, "Multi-task learning to improve egocentric action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[55] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

[56] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, PMLR, 2015.

[57] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1426–1435, 2019.

[58] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," *Advances in neural information processing systems*, vol. 31, 2018.

[59] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5400–5409, 2018.

[60] Q. Dou, D. Coelho de Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," *Advances in neural information processing systems*, vol. 32, 2019.

[61] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 624–639, 2018.

[62] S. Bucci, A. D'Innocente, Y. Liao, F. M. Carlucci, B. Caputo, and T. Tommasi, "Self-supervised learning across domains," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5516–5528, 2021.

[63] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2229–2238, 2019.

[64] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[65] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The epic-kitchens dataset," in *European Conference on Computer Vision (ECCV)*, 2018.

[66] M. Lu, D. Liao, and Z.-N. Li, "Learning spatiotemporal attention for egocentric action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[67] K. Min and J. J. Corso, "Integrating human gaze into attention for egocentric activity recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1069–1078, 2021.

[68] Y. Cheng, F. Fang, and Y. Sun, "Team vi-i2r technical report on epic-kitchens-100 unsupervised domain adaptation challenge for action recognition 2021," *arXiv preprint arXiv:2206.02573*, 2022.

[69] Y. Zhang, H. Doughty, L. Shao, and C. G. Snoek, "Audio-adaptive activity recognition across video domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13791–13800, 2022.

[70] A. Nasirimajd, S. A. Peirone, C. Plizzari, and B. Caputo, "Epic-kitchens-100 unsupervised domain adaptation challenge: Mixed sequences prediction," *arXiv preprint arXiv:2307.12837*, 2023.

[71] M. Planamente, G. Goletto, G. Trivigno, G. Averta, and B. Caputo, "Polito-iit-cini submission to the epic-kitchens-100 unsupervised domain adaptation challenge for action recognition," *arXiv preprint arXiv:2209.04525*, 2022.

[72] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[73] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.