Comparative Study of Deep Learning Architectures for Sports Activity Recognition

1st Sharva Khandagale
Robotics
University of Minnesota
khand137@umn.edu

***Abstract*—Activity recognition plays a pivotal role in various practical applications, and sports and fitness are key domains that benefit from its use. In this context, accurate performance analysis, fitness tracking, and automated coaching are among the primary use cases. Despite the availability of numerous deep learning models for activity recognition, there is an ongoing debate about the optimal model for specific applications. This study provides a comparative analysis of several deep learning architectures for sports activity recognition, evaluating models based on key performance metrics such as training and validation loss, as well as accuracy. The focus of this project is on sports-related activities, aiming to offer insights into the strengths and weaknesses of different models and guide the selection of the most suitable architectures for real-world sports applications. Among all the models evaluated, the Vision Transformer emerged as the best, achieving a validation loss of 0.046 and an accuracy of 99.2%.**

***Index Terms*—CNN, ResNet, ConvXTiny, MobileNet, Vision Transformer, Deep Learning,**

I. Introduction

In recent years, activity recognition has gained significant attention due to its diverse applications in areas such as surveillance and security, where it aids in detecting suspicious activities, crowd monitoring, fall detection, and more, as well as in sports and fitness, where it can support automated coaching. For example, automatically tagging training videos with the correct sport label makes it easier to search, filter, and index sports content, facilitating tracking of training progress. Previous research has explored various methods, including CNNs, RNNs, and LSTMs. However, each approach has its own advantages and limitations, making it more suitable for some applications and less effective for others. Selecting the right deep learning architecture for activity recognition is critical, as it directly impacts the system's accuracy, efficiency, and ability to manage the complexity of the task. Several factors must be considered when choosing the appropriate architecture, including the following :

1) **Data Complexity**: The type and volume of data significantly influence architecture selection. For example, when working with images or videos (e.g., from cameras), 2D or 3D CNNs are more suitable due to their capacity to capture spatial and spatial-temporal features.
2) **Task Variability**: Activity recognition tasks can range from simple activities (e.g., walking, running) to complex ones (e.g., cooking, exercising). The chosen architecture must be flexible enough to handle such variability. Simple activities may require less complex models, while more complex activities may benefit from multimodal architectures, such as combining sensor data with video inputs.
3) **Accuracy and Robustness**: Certain architectures are more robust to noise and data variability. For example, RNNs and LSTMs are better suited for handling noisy or missing data in sequential tasks due to their memory capabilities, whereas CNNs perform well with structured data such as images, but may require additional preprocessing or hybrid models for unstructured data, such as sensor readings.

According to existing literature, CNNs and transformer-based models have proven successful for activity recognition tasks. Motivated by the potential of selecting the most appropriate architecture to significantly enhance performance, this paper presents a comparative study of several deep learning models, specifically targeting sports-related activities. The goal is to identify the most effective architecture for real-world applications, including ResNet, MobileNet, ConvXTiny and Vision Transformer. The main contributions of this paper are: (1) a detailed evaluation of multiple deep learning models based on performance metrics such as accuracy, validation loss, and computational efficiency, and (2) a thorough analysis of the strengths and weaknesses of each model in the context of sports activity recognition. The rest of the paper is structured as follows: Section II outlines the working of various Deep Learning architectures, Section III presents the experimental results, and Section IV concludes the paper.

II. Working of Models

A. ResNet50

ResNet50, a deep variant of Residual Networks introduced by He [2], addresses the challenges associated with training very deep neural networks, such as the vanishing gradient problem, by leveraging residual connections.

1) Residual Blocks and Gradient Flow: ResNet50 uses residual blocks, which incorporate skip connections. These

allow the input to bypass certain layers and be added directly to the output of a residual function. Mathematically, each residual block can be expressed as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{1}$$

Here, $\mathbf{x}$ is the input, $\mathcal{F}(\mathbf{x}, \{W_i\})$ represents the residual function composed of convolutional layers with weights W_i, and $\mathbf{y}$ is the output.

2) Weight Initialization and Optimization: The model typically employs He (Kaiming) initialization, which sets the weights based on the number of input units to maintain the variance of activations across layers:

$$W \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \tag{2}$$

Weight decay acts as an L2 regularization term, preventing the weights from growing too large and helping to avoid overfitting.

3) L2 Regularization (Ridge): L2 regularization adds the squared magnitude of weights to the loss function:

$$L_{\text{total}} = L_{\text{task}} + \lambda \sum_i W_i^2 \tag{3}$$

This encourages the model to keep the weights small and evenly distributed, which helps in preventing overfitting and ensures numerical stability during training.

4) L1 Regularization (Lasso): L1 regularization adds the absolute value of weights to the loss function:

$$L_{\text{total}} = L_{\text{task}} + \lambda \sum_i |W_i| \tag{4}$$

L1 regularization promotes sparsity in the weights, effectively performing feature selection by driving some weights to zero.

5) Combined Regularization (Elastic Net): For a balanced approach, Elastic Net combines both L1 and L2 regularization:

$$L_{\text{total}} = L_{\text{task}} + \lambda_1 \sum_i |W_i| + \lambda_2 \sum_i W_i^2 \tag{5}$$

This allows the model to benefit from both weight shrinkage and sparsity, providing a more flexible regularization strategy.

B. MobileNet V2

Its design emphasizes efficiency and accuracy, making it ideal for resource-constrained environments.

1) Gradient Dynamics in MobileNetV2: MobileNetV2 employs *inverted residuals* and *linear bottlenecks*, which play crucial roles in maintaining effective gradient flow.

a) Inverted Residuals and Gradient Flow: MobileNetV2 incorporates residual connections similar to those in ResNet architectures, addressing the vanishing gradient problem The residual connections in MobileNetV2 are mathematically represented as:

$$\mathbf{y} = \mathbf{x} + \mathcal{F}(\mathbf{x}, \mathbf{W}) \tag{6}$$

where $\mathbf{x}$ is the input, $\mathcal{F}$ denotes the residual function (comprising depthwise separable convolutions), and $\mathbf{W}$ represents the weights.

During backpropagation, the gradient with respect to the input $\mathbf{x}$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \left(\mathbf{I} + \frac{\partial \mathcal{F}}{\partial \mathbf{x}}\right) \tag{7}$$

Here, $\mathbf{I}$ is the identity matrix, indicating that the gradient is a sum of the direct gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ and the gradient flowing through the residual function. This additive property preserves gradient magnitude across layers, facilitating stable and efficient training.

b) Activation Functions and Gradient Behavior: MobileNetV2 utilizes the *ReLU6* activation function, a variant of the Rectified Linear Unit (ReLU) that caps activations at 6. The ReLU6 function is defined as:

$$\text{ReLU6}(x) = \min(\max(0, x), 6) \tag{8}$$

This bounded activation function helps prevent the exploding gradient problem by limiting the maximum activation value, thereby contributing to more stable gradient updates during training.

2) Weight Structure and Parameter Efficiency: The weight parameters in MobileNetV2 are strategically structured to balance model complexity and computational efficiency. The architecture leverages *depthwise separable convolutions*, decomposing standard convolutions into depthwise and pointwise (1x1) convolutions.

a) Depthwise Separable Convolutions: A standard convolution with K filters of size $D \times D$ applied to an input with C channels has a computational cost of $\mathcal{O}(C \times D \times D \times K)$. In contrast, a depthwise separable convolution splits this into:

1) **Depthwise Convolution**: Applies a single $D \times D$ filter per input channel, resulting in $\mathcal{O}(C \times D \times D)$ computations.
2) **Pointwise Convolution**: Utilizes 1×1 convolutions to combine the outputs of the depthwise convolution, incurring $\mathcal{O}(C \times K)$ computations.

Overall, depthwise separable convolutions reduce the computational complexity from $\mathcal{O}(C \times D \times D \times K)$ to $\mathcal{O}(C \times D \times D + C \times K)$, enhancing efficiency without significantly compromising representational capacity.

b) Linear Bottlenecks: The linear bottleneck layers in MobileNetV2 restrict the dimensionality of feature maps, reducing the number of parameters and mitigating overfitting. Mathematically, if $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{out}}$ represent the weights of the input and output layers of the bottleneck, the transformation is expressed as:

$$\mathbf{y} = \mathbf{W}_{\text{out}} \cdot \sigma(\mathbf{W}_{\text{in}} \cdot \mathbf{x}) \tag{9}$$

where σ denotes a non-linear activation function (e.g., ReLU6). The linear bottleneck ensures that the output has reduced dimensionality, promoting parameter efficiency.

3) Regularization via ℓ_1 and ℓ_2 Norms:

a) ℓ_1 Norm Regularization: The ℓ_1 norm regularization adds a penalty proportional to the absolute value of the weights:

$$\mathcal{L}_{\ell_1} = \lambda \sum_i |w_i| \tag{10}$$

where λ is the regularization coefficient. This form of regularization encourages sparsity in the weight parameters, potentially leading to models that are more interpretable and efficient by effectively zeroing out less important weights.

b) ℓ_2 Norm Regularization: Alternatively, the ℓ_2 norm regularization imposes a penalty proportional to the square of the weights:

$$\mathcal{L}_{\ell_2} = \lambda \sum_i w_i^2 \tag{11}$$

ℓ_2 regularization discourages large weights by distributing the error across all parameters, thus promoting weight decay and enhancing the model's ability to generalize to unseen data.

c) Application in MobileNetV2: In the context of MobileNetV2, applying ℓ_1 and ℓ_2 regularization serves multiple purposes:

1) **Weight Decay**: By penalizing large weights, the model avoids overfitting, ensuring that it captures the underlying data distribution rather than memorizing training samples.
2) **Sparsity Promotion**: ℓ_1 regularization can induce sparsity, which is beneficial for deploying models on devices with limited computational resources.
3) **Balanced Regularization**: Combining ℓ_1 and ℓ_2 norms (Elastic Net regularization) leverages the benefits of both sparsity and weight decay, providing a balanced approach to regularization.

The overall loss function incorporating both regularization terms is expressed as:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_1 \mathcal{L}_{\ell_1} + \lambda_2 \mathcal{L}_{\ell_2} \tag{12}$$

where $\mathcal{L}_{\text{task}}$ represents the primary loss (e.g., cross-entropy), and λ_1 and λ_2 are the respective regularization coefficients.

4) Conclusion:

C. ConvXTiny

The *ConvXTiny* model, introduced in [3], represents a streamlined variant within the ConvNeXt V2 architecture, optimized for scenarios requiring lightweight yet effective convolutional neural networks (CNNs).

1) Gradient Dynamics in ConvXTiny: Mathematically, the gradient of the loss function $\mathcal{L}$ with respect to a weight parameter w is expressed as:

$$\frac{\partial \mathcal{L}}{\partial w} = \delta \cdot x, \tag{13}$$

where δ denotes the error term propagated from subsequent layers, and x represents the input activation to the neuron associated with weight w. The reduced number of parameters in ConvXTiny not only accelerates gradient computations but also mitigates issues related to vanishing or exploding gradients, thereby promoting stable and efficient training dynamics.

2) Weight Initialization and Optimization Strategies: Effective weight initialization is pivotal for the convergence and performance of ConvXTiny. The model employs initialization techniques which are designed to maintain the variance of activations throughout the network. Specifically, weights are initialized as:

$$w \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right), \tag{14}$$

where n_{in} is the number of input units to the layer. This strategy facilitates symmetry breaking and ensures robust gradient flow during training.

3) Regularization Techniques: L1 and L2 Norms: To prevent overfitting and enhance the generalization of ConvXTiny, both L1 and L2 regularization norms are integrated into the training process. These regularization techniques impose penalties on the weight parameters, encouraging the model to learn simpler and more generalizable patterns.

a) L1 Regularization: The L1 norm is defined as:

$$\|W\|_1 = \sum_i |w_i|, \tag{15}$$

where W represents the weight matrix. L1 regularization promotes sparsity by driving many weights to zero, effectively performing feature selection and simplifying the model. This sparsity not only reduces the model's complexity but also enhances interpretability by highlighting the most significant features.

b) L2 Regularization: The L2 norm is given by:

$$\|W\|_2^2 = \sum_i w_i^2, \tag{16}$$

which penalizes large weights by considering their squared values. L2 regularization discourages the model from relying excessively on any single feature, promoting a more distributed and stable weight distribution. This leads to smoother decision boundaries and improves the model's resilience to noise in the training data.

4) Integrated Regularization in ConvXTiny: In the ConvX-Tiny model, the total loss function incorporates both L1 and L2 regularization terms :

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda_1 \|W\|_1 + \lambda_2 \|W\|_2^2, \tag{17}$$

where λ_1 and λ_2 are hyperparameters that control the strength of L1 and L2 regularizations, respectively.

5) Mathematical Insights and Practical Implications: The modified loss surface, influenced by L1 and L2 penalties, guides the optimization process toward solutions that are both sparse and stable. This dual regularization not only simplifies the model by reducing unnecessary weights but also strengthens its ability to generalize to unseen data.

Mathematically, the constraints imposed by the regularization terms limit the feasible parameter space, directing the optimization towards regions that favor simpler models with better generalization properties.

D. Vision Transformer (ViT)

Vision Transformers (ViTs) have revolutionized the field of image recognition by leveraging the transformer architecture's capacity to model long-range dependencies within image data [1].

1) Weight Initialization and Optimization: During training, these weights are updated iteratively to minimize the loss function $\mathcal{L}$ using gradient-based optimization algorithms like Adam [1]:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \cdot \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}^{(t)}), \tag{18}$$

where η represents the learning rate, and $\nabla_{\mathbf{W}} \mathcal{L}$ denotes the gradient of the loss with respect to the weights at iteration t.

2) Gradient Flow and Backpropagation: Backpropagation computes gradients through the chain rule, enabling the adjustment of weights based on their contribution to the final loss. For a weight matrix $\mathbf{W}^{(l)}$ in layer l, the gradient is computed as:

$$\nabla_{\mathbf{W}^{(l)}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}}, \tag{19}$$

where $\mathbf{z}^{(l)}$ represents the pre-activation output of layer l. To prevent issues such as vanishing or exploding gradients, ViTs incorporate layer normalization and residual connections, which stabilize the gradient flow and facilitate the training of deep networks.

3) Regularization with L2 Norm: The ViT employs L2 norm regularization, also known as weight decay, to constrain the magnitude of the weights:

$$\mathcal{L}_{\text{L2}} = \lambda \sum_i W_i^2, \tag{20}$$

where λ is the regularization coefficient and W_i denotes individual weight parameters.

In the context of gradient-based optimization, L2 regularization modifies the weight update rule as follows:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \left(\nabla_{\mathbf{W}} \mathcal{L} + 2\lambda \mathbf{W}^{(t)} \right). \tag{21}$$

This adjustment effectively applies a penalty that shrinks the weights proportionally to their current values, promoting weight decay and mitigating overfitting.

4) Mathematical Insights: The inclusion of L2 regularization introduces a trade-off between minimizing the training loss and keeping the model weights small. The regularized loss function can be expressed as:

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \mathcal{L}_{\text{L2}} = \mathcal{L} + \lambda \sum_i W_i^2. \tag{22}$$

Taking the gradient with respect to a weight W_i, we obtain:

$$\frac{\partial \mathcal{L}_{\text{total}}}{\partial W_i} = \frac{\partial \mathcal{L}}{\partial W_i} + 2\lambda W_i. \tag{23}$$

This demonstrates that each weight update is influenced by the gradient of the loss as well as its current magnitude, enforcing a form of regularization that promotes simpler models with better generalization capabilities.

III. Data Processing and Model Creation

The dataset contains 13,429 images in training dataset and 500 images in validation dataset. All the images are resized to be 224 x 224 x 3 in each model and augmentation is introduced to increase the robustness of the models. There are 100 classes in the dataset and all the models have been modified to accept this specific number of classes by adding few additional layers to the original model. The weights for ResNet, MobileNet and ConvXTiny are 'imagenet' weights available in deep learning frameworks and libraries. Having same weights introduces similar starting points for models upon which the experiments are conducted. The ViT model being a transformer architecture different from neural networks has it's own base weights.

IV. Experimental Results

A. Performance of ResNet50

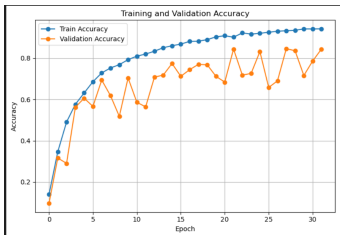


Fig. 1: ResNet50: Training and Validation Accuracy

The model does overfitting which is seen through divergence in validation. Significant variations in L1 norms, in later layers, might signal over-parameterization in deeper layers. Regularization techniques such as dropout, weight decay can help in improving the performance of the model. Although data augmentation was implemented, the model needs layer-by-layer tuning of L1 and L2 norms.

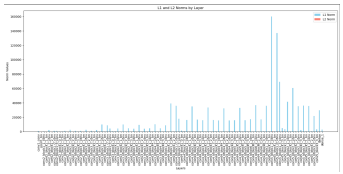


Fig. 2: ResNet50: L1 and L2 Norms in layers

B. Perfromance of MobileNet V2

Fig. 3: MobileNet V2: Training and Validation Loss

The model performs better in validation tests than training tests. There might be underfitting based on graph data and other data obtained from training in model. Fine-tuning of hyperparameters (learning rate, batch size) or optimization methods could be beneficial.

C. Performance of ConvXTiny

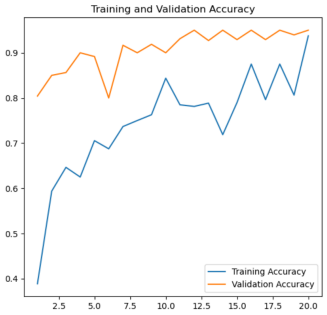


Fig. 4: ConvXTiny: Training and Validation Accuracy

Training and validation accuracy show consistent improvement, reaching above 90% validation accuracy.The convergence pattern implies the model generalizes well to unseen data with minimal overfitting. However, parameters like number of epochs, learning rate or adjusting deeper layers can be factors which can improve the estimated performance of the model to over 95%.

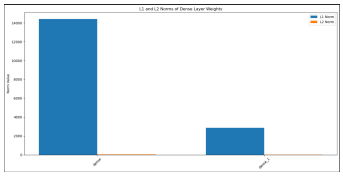
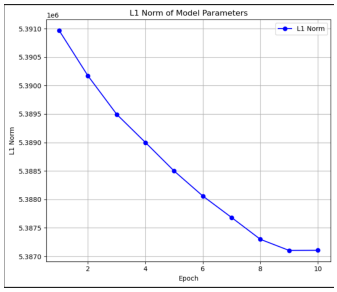


Fig. 5: ConvXTiny: Norm in dense layers

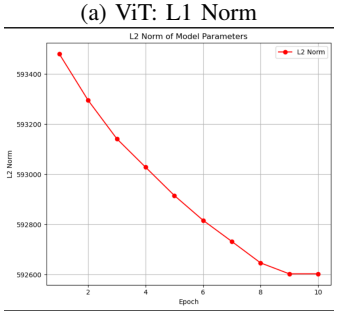
D. Performance of ViT (Vision Transformer)

Fig. 6: Training and Validation accuracy of ViT

The performance evaluation of the Vision Transformer (ViT) model compares training and validation accuracy, both rapidly improving and stabilizing near 100 %, highlighting excellent generalization. The L1 and L2 norm show a steady decline over epochs, indicating effective weight optimization and convergence. The model shows 99.2% accuracy which is the best outcome of all the architectures experimented on.



(a) ViT: L1 Norm



(b) ViT: L2 Norm

ACKNOWLEDGMENT

The dataset is made available at Kaggle by Gerry with the username gpiosenka. The data set contains no duplicate images and has been carefully crafted to be of high quality. The quality of data is crucial to machine learning and part of bigger success is attributed to this.

REFERENCES

[1] https://arxiv.org/abs/2010.11929, An Image is worth 16x16 Words
[2] https://arxiv.org/abs/1512.03385, Deep Residual Learning for Image Recognition
[3] https://arxiv.org/abs/2301.00808, ConvNeXt V2
[4] https://arxiv.org/abs/1801.04381, MobileNet V2