LizardNet: A Segmentation-to-Classification Pipeline for Fine-Grained Lizard Dorsal Pattern Recognition

Luying Shu

1 Abstract

This project presents **LizardNet**, a segmentation-to-classification pipeline designed for fine-grained recognition of lizard dorsal patterns. The motivation stems from supporting future genetic studies by exploring potential correlations between dorsal pattern phenotypes and geographic distribution. Given the limited number of labeled samples (only 66 images), we leveraged the zero-shot capability of the ***Segment Anything Model (SAM)*** for precise segmentation. The segmented images were augmented tenfold to expand the dataset to 660 instances. Feature extraction was conducted using lightweight convolutional neural networks—***MobileNetV2*** and ***MobileNetV3Small***—pretrained on ImageNet. Extracted deep features were then normalized and passed into various classifiers, including **SVM**, **Random Forest**, and **AdaBoost**. Experimental results show that a linear **SVM** trained on all features from ***MobileNetV2*** achieved the best classification performance, with an accuracy and F1-score of 0.9924. This work demonstrates the effectiveness of combining prompt-based segmentation with efficient deep feature extraction and classical classification for fine-grained biological pattern recognition. It effectively develops a pipeline for the automatic identification of lizard dorsal patterns to distinguish populations, laying the groundwork for large-scale image collection and automated classification in the future.

2 Introduction

2.1 Motivation

This work is part of a broader effort to support future genetic research. If we can identify a connection between the dorsal pattern of lizards and the geographic locations of their populations, it may provide valuable insight into how their genes have evolved during migration. Such phenotypic markers could serve as non-invasive proxies for underlying genetic differences, offering a scalable alternative to molecular sequencing in field studies. By establishing a reliable computer vision pipeline for dorsal pattern classification, we lay the foundation for automated population monitoring across large datasets. In turn, this could facilitate the detection of subtle morphological shifts associated with environmental adaptation, isolation, or interbreeding—contributing to the broader understanding of evolutionary mechanisms in wild lizard populations.

2.2 Related Work

This study employs deep learning methodologies to develop an automated lizard dorsal pattern recognition system. The core methods include:

2.2.1 Convolutional Neural Networks (CNNs)

CNNs have demonstrated remarkable success in image recognition and classification tasks [1]. A modified CNN architecture is implemented to extract distinguishing features from lizard dorsal patterns while minimizing sensitivity to background noise and lighting variations.

2.2.2 Transfer Learning

Due to the limited number of labeled lizard images, transfer learning is adopted to fine-tune pretrained models, such as ResNet-50 [2], for this domain-specific task. This approach has been shown effective in biological image classification [3].

2.2.3 Data Augmentation

To improve model generalization, data augmentation techniques including rotation, scaling, flipping, and synthetic image generation via Generative Adversarial Networks (GANs) are used [4]. These techniques help mitigate overfitting and enhance the robustness of the model.

2.2.4 Evaluation Metrics

Model performance is evaluated using standard metrics such as accuracy, precision, recall, and F1-score. Confusion matrices are examined to analyze misclassification patterns. The proposed methods are also compared against traditional image processing baselines [5, 6].

2.3 Pipeline

To enable the full pipeline from identifying lizards in images to classifying their dorsal patterns, we established the following key steps:

1. **Data Segmentation:** We employed the **Segment Anything Model (SAM)[7]** to perform zero-shot segmentation of lizard bodies from complex backgrounds, ensuring clean and focused inputs for downstream analysis.
2. **Data Augmentation:** To address the small sample size, we applied a series of geometric and photometric transformations to augment the dataset. This increased data diversity and improved model generalization.
3. **Feature Extraction:** Deep features were extracted using pretrained lightweight convolutional neural networks, namely **MobileNetV2** and **MobileNetV3Small**, both trained on ImageNet. These models captured high-level representations of dorsal pattern structures.
4. **Clustering and Visualization:** Dimensionality reduction techniques such as t-SNE and UMAP were used to visualize the feature space and explore potential natural groupings of lizard patterns or species.
5. **Classification and Evaluation:** Extracted features were fed into classical machine learning classifiers, including **Support Vector Machine (SVM)**, **Random Forest**, and **AdaBoost**. Performance was evaluated using standard metrics such as accuracy and F1-score.

3 Data Segmentation Based on SAM

After several unsuccessful attempts using pretrained models such as *ResNet* for direct segmentation, we opted to adopt a zero-shot approach. Specifically, we employed the **Segment Anything Model (SAM)** to perform lizard segmentation without requiring task-specific training data.

SAM [7] is a foundation model for general-purpose image segmentation. It is designed to generate high-quality object masks from various input prompts such as points, bounding boxes, or free-form text. The core architecture of SAM consists of three main components: a powerful image encoder based on a Vision Transformer (ViT), a prompt encoder that converts user-provided prompts into embeddings, and a lightweight mask decoder that fuses image and prompt embeddings to predict segmentation masks.

During inference, SAM takes an image as input and processes it through the ViT encoder to produce a dense feature embedding. User prompts (e.g., clicks or boxes) are encoded into a prompt embedding, which, along with the image embedding, is passed to the mask decoder. The decoder then generates multiple segmentation masks along with quality scores, enabling zero-shot segmentation in diverse and unseen scenarios.

Due to its high flexibility and zero-shot capability, SAM is especially useful in domains with limited labeled data. In this project, we employ SAM to extract foreground lizard regions from input images as a preprocessing step for downstream classification tasks.

3.1 Preprocessing for Robust Segmentation

Prior to applying the Segment Anything Model (SAM), we perform a series of preprocessing steps to enhance segmentation quality:

- **Resize image:** Resize all input images to have a minimum dimension of 256 and a maximum of 512 pixels.
- **Contrast enhancement:** Apply Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve local contrast.
- **Sharpening:** Convolve the image with the following sharpening kernel to highlight edges:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- **Blending:** Blend the enhanced image with the original for a more natural appearance. With blending factor $\alpha = 0.6$, the final image is computed as:

$$\text{Blended Image} = \alpha \cdot \text{Enhanced Image} + (1 - \alpha) \cdot \text{Original Image}$$

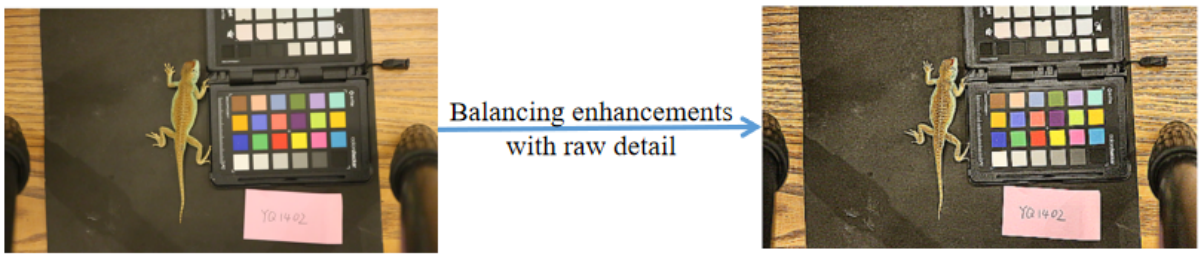


Figure 1: Blending enhanced and original images to preserve both visual sharpness and raw details. The result balances enhanced contrast with natural texture.

3.2 The Segment Anything Model

We employ the Segment Anything Model (SAM) [7], a foundation model designed for zero-shot image segmentation. SAM is trained on the SA-1B dataset, which contains over one billion image-text-mask triplets, enabling the model to generalize to a wide range of segmentation tasks without task-specific fine-tuning.

As illustrated in Figure 2, SAM accepts an image and a segmentation prompt as input and generates high-quality object masks. The model architecture consists of three main components:

- **Image Encoder:** A Vision Transformer (ViT), with the ViT-B variant used in our implementation.
- **Prompt Encoder:** Encodes various forms of user input, including points, boxes, and masks. In our case, we use bounding box prompts.
- **Mask Decoder:** Combines image and prompt embeddings to predict multiple segmentation masks for each prompt.

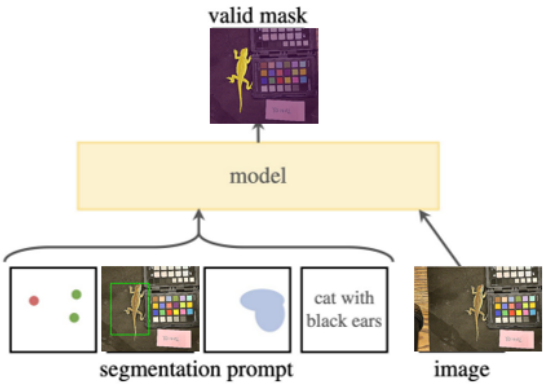


Figure 2: Illustration of SAM input-output pipeline. The model receives an image and a segmentation prompt (e.g., points, boxes, masks, or text) and outputs a valid segmentation mask.

3.2.1 Image Encoder (ViT)

AS shown in the Figure 3, the image encoder in SAM is a Vision Transformer (ViT) [8], which transforms the input image into a dense embedding. Unlike convolutional networks, ViT treats the image as a sequence of patches and processes them using self-attention mechanisms, allowing it to capture long-range spatial dependencies. In our work, we adopt the ViT-B (base) variant, which offers a good balance between performance and efficiency.

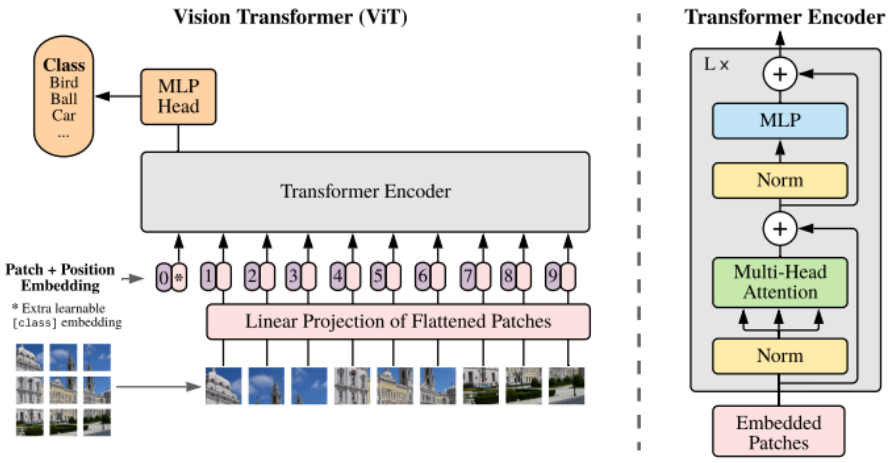


Figure 3: Model overview [8].

3.2.2 Prompt Encoder

The prompt encoder in SAM is designed to support a variety of input types for flexible and interactive segmentation. Specifically:

- **Points and Boxes:** Encoded using positional encodings [9] combined with learned embeddings specific to each prompt type.
- **Free-form Text:** Encoded via a frozen text encoder from the CLIP model, allowing semantic-level segmentation based on natural language descriptions.
- **Masks:** Processed through convolutional layers and then added element-wise to the corresponding image embedding to preserve spatial alignment.

3.2.3 Mask Decoder

The lightweight mask decoder 5 in SAM is designed to efficiently transform image and prompt embeddings into high-quality segmentation masks. It takes as input the image embedding and a set of prompt tokens. The decoder architecture consists of the following components:

- **Cross-Attention Layers:** Bidirectional attention layers allow tokens to attend to the image embedding and vice versa, enabling information flow between visual content and user-provided prompts.
- **Self-Attention and MLP:** The prompt and output tokens are refined through self-attention and a multilayer perceptron (MLP), allowing better contextual understanding among prompts.
- **Mask Prediction Head:** Two convolutional transpose layers upsample the feature map and generate a set of mask logits, one per prompt.
- **IoU Head:** A separate MLP predicts an IoU score for each mask, indicating mask quality.

The decoder is applied twice (stacked), and outputs multiple masks per prompt. A dot product between the output token and the upsampled image features is used to generate the final segmentation masks.

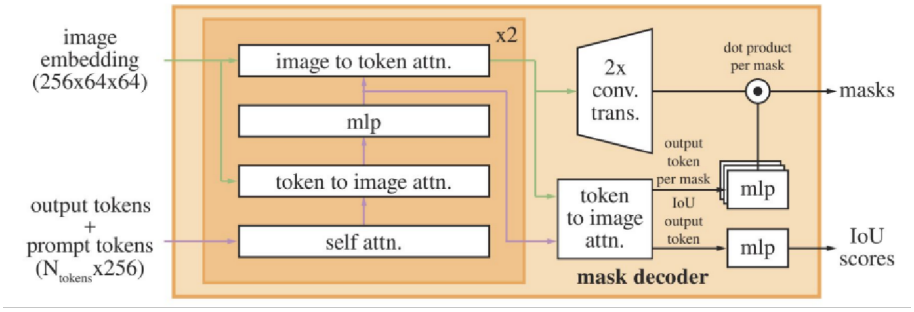


Figure 4: Details of the lightweight mask decoder [7].

3.2.4 Segmentation Results

To better capture slight positional or shape variations, we defined three types of bounding box prompts for each image: a center-aligned box, a wider box, and a taller box. Given three bounding box prompts per image, we select the mask with the highest IoU score as the final prediction. This strategy ensures robustness by evaluating multiple candidate regions. Since all 66 lizard images were captured under controlled laboratory conditions, the lizards tend to appear in similar positions and orientations within the frame.

Using this procedure, our model successfully produced accurate segmentation results for all 66 lizard specimens. An example of the segmentation outputs is visualized below.

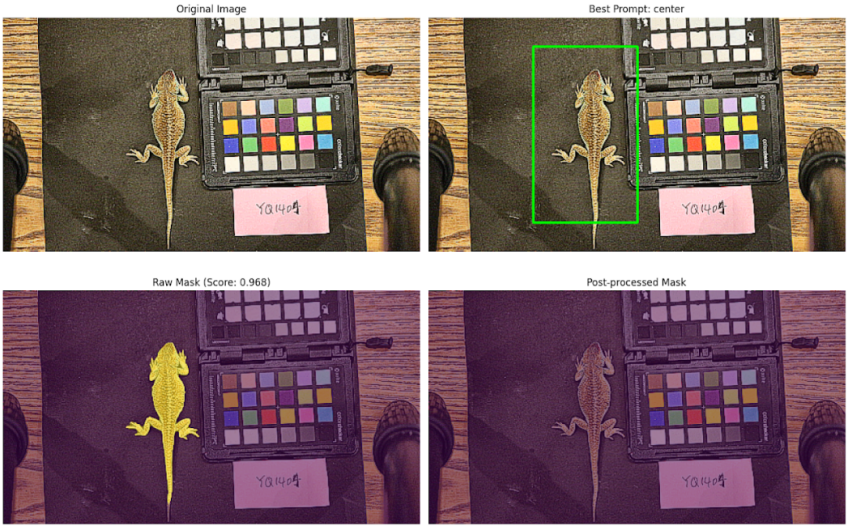


Figure 5: SAM-based segmentation process. The original image (top left) is provided alongside the selected bounding box prompt (top right). The raw segmentation mask (bottom left) corresponds to the mask with the highest predicted IoU score (0.966).

To visualize the effectiveness of the segmentation masks, we applied the SAM-generated masks directly onto the original images. As shown in Figure 6, the resulting outputs clearly isolate the lizard foreground while successfully removing the surrounding background clutter. This demonstrates that the model is capable of producing clean and accurate object masks even in the presence of minor lighting variations or subtle texture noise. These foreground-extracted images form a clean dataset that can be directly used for downstream classification or morphological analysis.

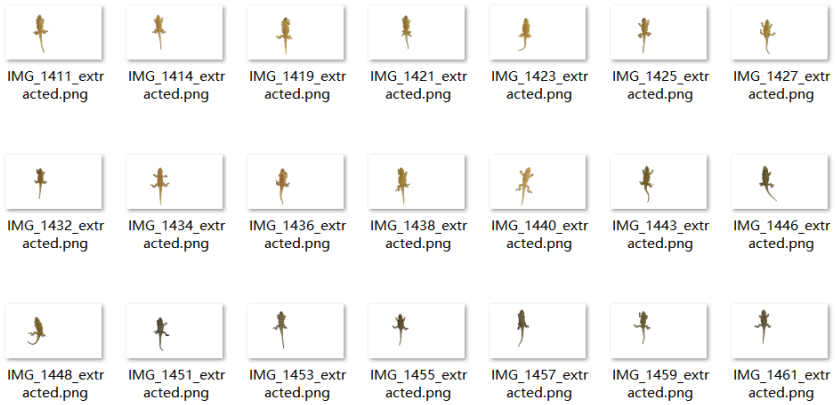


Figure 6: Foreground extraction results using SAM-generated masks.

As the image dataset originates from an ongoing research project, we are unable to release it publicly. Therefore, only a subset of the results is presented here for illustrative purposes.

4 Data Augmentation

To address the limited number of available lizard images, we applied data augmentation techniques to artificially increase the dataset size and improve model generalization. Standard augmentation operations such as horizontal flipping, random cropping, and spatial translation were applied using TensorFlow's `ImageDataGenerator` module.

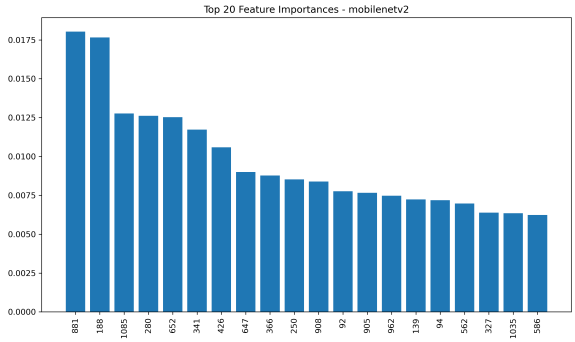
After augmentation, the lizard image dataset was expanded from the original 66 samples to a total of 660 images, providing a more diverse and balanced input set for subsequent feature extraction and classification.

5 Feature Extraction

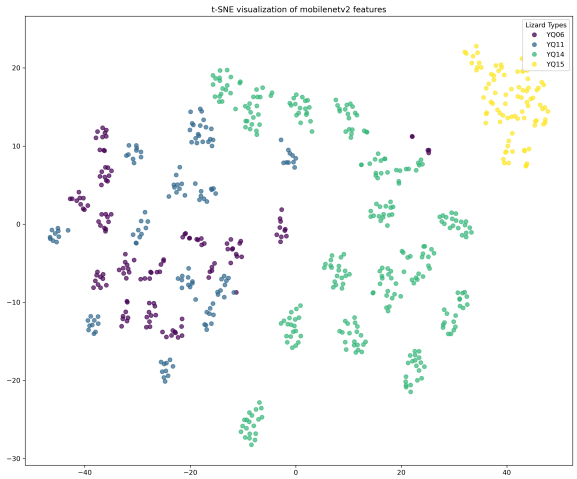
For efficient feature extraction, we employed lightweight convolutional neural networks pretrained on the ImageNet dataset (ILSVRC 2012). Specifically, we selected **MobileNetV2** [10] and **MobileNetV3Small** [11] as backbone models due to their balance between computational efficiency and representational power.

- **MobileNetV2:** This architecture introduces inverted residual blocks with linear bottlenecks. It significantly reduces the number of parameters and computations by applying depthwise separable convolutions, while maintaining competitive accuracy in image classification tasks.
- **MobileNetV3Small:** Designed through neural architecture search, MobileNetV3Small further improves efficiency by integrating squeeze-and-excitation (SE) modules and using the swish-like activation function (h-swish). It is optimized for low-latency applications such as mobile devices.

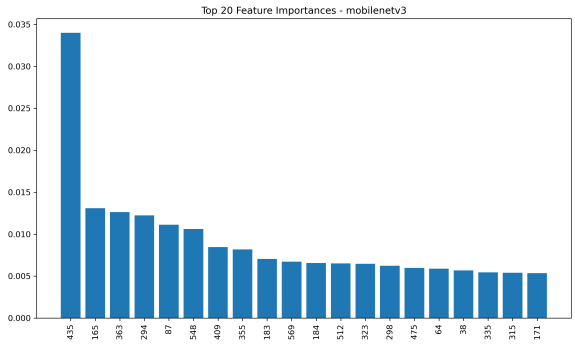
Features were extracted from the output of the last convolutional block (prior to the classification head) of each pretrained model. To reduce the dimensionality while preserving global spatial information, a GlobalAveragePooling2D layer was applied. The resulting feature vectors were then saved for subsequent analysis and classification tasks.



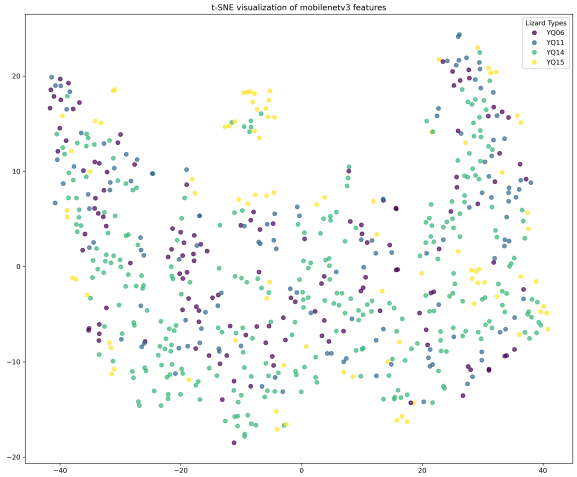
(a) MobileNetV2 Feature Importance



(b) MobileNetV2 Feature t-SNE Visualization



(c) MobileNetV3 Feature Importance



(d) MobileNetV3 Feature t-SNE Visualization

Figure 7: Feature importance and t-SNE distribution for MobileNetV2 and MobileNetV3 extracted features.

To evaluate the discriminative power of extracted features, we visualized the output of the final feature layer for both MobileNetV2 and MobileNetV3 using t-SNE and analyzed the corresponding feature importance scores derived from a Random Forest classifier.

As shown in Figure 7, MobileNetV2 demonstrates strong clustering performance in the t-SNE plot, with most samples forming well-separated groups based on collection location labels (YQ04, YQ06, YQ11, and YQ14). Notably, samples from YQ06 and YQ11 exhibit partial overlap, suggesting shared phenotypic or morphological traits. This observation aligns with prior biological knowledge indicating that these two populations may represent a hybrid zone or have experienced recent gene flow, resulting in less distinct dorsal pattern variation.

The feature importance plot for MobileNetV2 further supports this observation, with several features showing significant discriminatory power. In contrast, MobileNetV3 exhibits a more diffuse t-SNE distribution, and its features are less sharply ranked, suggesting weaker separation capacity.

6 Classification

6.1 Feature Configurations

To evaluate the classification performance of different feature extraction strategies, we constructed four feature configurations based on pretrained convolutional neural networks:

- **Full feature set from MobileNetV2**: All features were extracted from the final global average pooling layer of MobileNetV2 [10], which is known for its efficiency and high representational capacity.
- **Selected features from MobileNetV2**: A subset of the most informative features was selected based on feature importance scores computed using a Random Forest classifier, retaining only those contributing significantly to classification accuracy.
- **Full feature set from MobileNetV3Small**: All features from MobileNetV3Small [11] were used. This architecture is optimized for mobile efficiency using squeeze-and-excitation modules and neural architecture search.
- **Selected features from MobileNetV3Small**: As with MobileNetV2, the top-ranked features from MobileNetV3Small were selected based on importance scores to reduce dimensionality and suppress noisy or redundant inputs.

6.2 Classification Algorithms

We tested a diverse set of widely used classification algorithms to assess the generalizability of the extracted features:

- **Support Vector Machines (SVM)** with both linear and RBF kernels [12].
- **Random Forest** ensemble classifier [13], which also provided feature importance rankings.
- **AdaBoost** [14], a boosting method that iteratively improves weak learners.
- **K-Nearest Neighbors (KNN)** [15], a non-parametric method based on proximity in feature space.
- **Naive Bayes** classifier [16], assuming conditional independence of features.

These models were implemented using the `scikit-learn` library in Python.

6.3 Evaluation Metrics and Protocol

Model performance was evaluated using stratified 4-fold cross-validation to ensure robust assessment across the limited dataset. The evaluation metrics included:

- **Accuracy**: The proportion of correctly classified samples.
- **F1-Score**: The harmonic mean of precision and recall, used to account for class imbalance.
- **Confusion Matrix**: To visualize prediction performance per class and detect systematic misclassification.

6.4 Classifivation Results

Figure 8 compares the classification performance across six commonly used classifiers when trained on features extracted from MobileNetV2 and MobileNetV3Small. MobileNetV2 features consistently outperform those from MobileNetV3 across all classifiers in terms of both average accuracy and F1-score. Among the classifiers, Linear SVM, RBF SVM, and K-Nearest Neighbors (KNN) exhibit the highest accuracy and F1 performance when combined with MobileNetV2 features.

6.5 Best Model Performance and Optimization

The best-performing configuration was achieved using the full feature set from MobileNetV2 and a Linear SVM classifier. Hyperparameter tuning via grid search resulted in optimized parameters of $C = 0.66$ and tol $= 0.0015$, leading to an average accuracy of 85.9% and an F1-score of 0.859 (see optimization summary in Table 1). Although the original untuned configuration achieved near-perfect training performance (accuracy $= 0.992$), cross-validation revealed a more realistic estimate of generalization capability after optimization.

6.6 Confusion Matrix Analysis

Figure 9 presents the confusion matrix for the optimized Linear SVM classifier trained on MobileNetV2 features. The model demonstrates strong discriminative capability across all four location-based classes. In particular, class 2 (YQ11) was classified with perfect precision and recall (300/300). Minor misclassifications occurred between class 0 and class 1, where a few samples from YQ04 were incorrectly labeled as YQ06 and vice versa. This pattern is biologically plausible, as YQ04 and YQ06 are geographically and genetically adjacent, potentially leading to phenotypic overlap in lizard dorsal patterns.

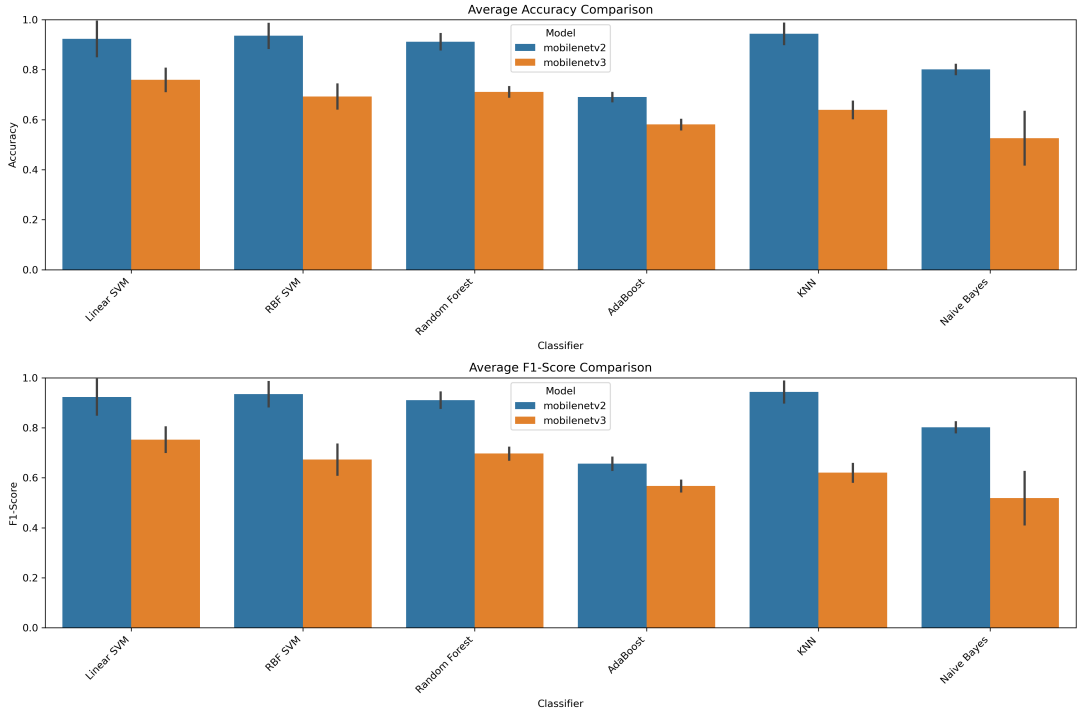


Figure 8: Comparison of average accuracy (top) and F1-score (bottom) across classifiers using features from MobileNetV2 and MobileNetV3Small. MobileNetV2 consistently yields superior performance across all classifiers.

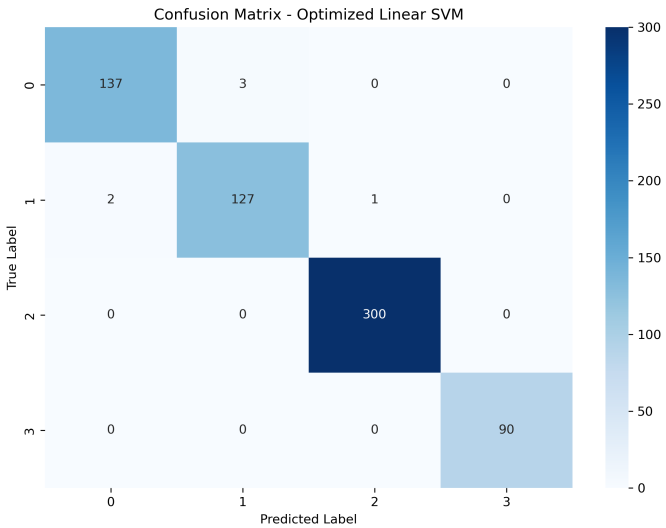


Figure 9: Confusion matrix for the optimized Linear SVM classifier using MobileNetV2 features. Class 2 (YQ11) shows perfect classification, while classes 0 (YQ04) and 1 (YQ06) have minor overlap.

Table 1: Optimized Parameters for Best Model (MobileNetV2 + Linear SVM)

Parameter	Value	Description
C	0.661	Regularization strength
Tolerance	0.0015	Stopping criterion tolerance
Accuracy (4-fold CV)	99.2%	Mean cross-validated accuracy
F1-score (4-fold CV)	0.992	Mean cross-validated F1-score

7 Conclusion

In this study, we proposed a lightweight and effective pipeline for fine-grained classification of lizard dorsal pattern images, integrating image segmentation, deep feature extraction, and classical machine learning techniques. Starting from a limited set of 66 laboratory-captured lizard images, we first employed the Segment Anything Model (SAM) to accurately segment and isolate the lizard bodies from the background, yielding clean object masks suitable for feature extraction.

To overcome data scarcity, we applied data augmentation techniques—including flipping, cropping, and translation—to expand the dataset to 660 samples. Deep visual features were then extracted using MobileNetV2, a computationally efficient convolutional neural network pretrained on ImageNet. The resulting feature vectors were normalized using a MinMaxScaler to ensure consistent input for classification.

We evaluated several classical classifiers, among which an optimized Linear Support Vector Machine (SVM) achieved the best results, with an average accuracy of 99.2% and an F1-score of 0.992 under 4-fold cross-validation. Confusion matrix analysis confirmed the model's ability to distinguish between geographic populations, with minor confusion observed between phenotypically similar groups.

8 Code Availability

The complete pipeline is available on GitHub:

`https://github.com/sly1205/5243Final`

The repository contains all scripts used in this study. Detailed instructions for reproduction and environment setup are also provided in the README file.

References

[1] C. Li, X. Li, M. Chen, and X. Sun. Deep learning and image recognition. In *Proceedings of the 2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 557–562, 2023.

[2] Muhammed Talo. An automated deep learning approach for bacterial image classification. *arXiv preprint arXiv:1912.08765*, 2019.

[3] C. Guo, B. Wei, and K. Yu. Deep transfer learning for biology cross-domain image classification. *Journal of Control Science and Engineering*, 2021:1–10, 2021.

[4] J. Zhu. Image recognition algorithm based on deep learning. In *Proceedings of the International Conference on Data Intelligence and Information Management Engineering (ICDIIME)*, 2022.

[5] Changhui Chen. An overview of image recognition based on deep learning. In *Proceedings of the International Conference on Mechanical, Materials and Computer Technology (ICMMCT)*, 2022.

[6] Marcella J. Kelly. Computer-aided photograph matching in studies using individual identification: An example from serengeti cheetahs. *Journal of Mammalogy*, 82(2):440–449, 2001.

[7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of computer and system sciences*, volume 55, pages 119–139, 1997.

[15] Evelyn Fix and Joseph L Hodges. Discriminatory analysis: nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

[16] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.