Hierarchical Insect Classification via DNA–Image Embedding Fusion

Mohammadamin Hajibagher Tehran (2080658)
Course: Deep Learning, 2025
Professor: Loris Nanni

August 29, 2025

Abstract

Identifying insect species is an important task in biology and ecology, but it's also very challenging because many species look similar and DNA alone doesn't always give the full picture. In this project, we built a multimodal deep learning model that combines both DNA sequences and images of insects to classify them more accurately. I fine-tuned a DNA encoder and a Vision Transformer (ViT) to create embeddings from each input, and then fused them together using an attention-based module. Instead of predicting the exact species in one step, the model uses a hierarchical approach: it first predicts the genus of the insect and then identifies the species within that genus. This design makes the classification task more efficient and reliable.

1 Introduction

Motivation. Insects are the most diverse group of organisms on Earth and play a vital role in ecosystems, agriculture, and biodiversity. Identifying insect species is challenging because many look similar, and DNA alone may not fully capture species differences.

Idea. We propose a multimodal deep learning model that combines DNA sequences and images. DNA is encoded using a fine-tuned Nucleotide Transformer, while images are processed with a fine-tuned Vision Transformer (ViT). Their embeddings, together with DNA sequence length information, are merged via an attention-based fusion module to form a unified representation. Exploiting taxonomy, the model predicts genus first and then species within that genus, reducing complexity and improving classification accuracy.

2 Related Works

Our approach is closely related to prior research on insect classification: *Advancing Taxonomy with Machine Learning: A Hybrid Ensemble for Species and Genus Classification* [1], and *Classifying the unknown: Insect identification with deep hierarchical Bayesian learning* [2]

2.1 Vision Transformers

The **Vision Transformer (ViT)** [3] introduced a novel way of applying the transformer architecture, originally designed for natural language processing, to computer vision tasks. Instead of using convolutional filters as in CNNs, ViT splits an image into fixed-size patches, embeds them as tokens, and processes them with self-attention. This design enables the model to capture both local patterns within patches and global dependencies across the entire image.

The model `google/vit-base-patch16-224` is one of the widely adopted pretrained ViT variants. It operates on 16×16 pixel patches from 224×224 images, providing a strong backbone for downstream fine-tuning tasks. ViTs have shown state-of-the-art performance in various classification problems and are particularly effective when large-scale pretraining is combined with fine-tuning on domain-specific data. In the context of insect identification, ViTs provide a way to automatically extract discriminative visual features such as wing patterns, body shape, and coloration, which are traditionally used by taxonomists.

2.2 Nucleotide Transformers

The rapid growth of genomic data has motivated the development of transformer-based models for biological sequences. The **Nucleotide Transformer** [4] is a large-scale model trained on diverse genomic datasets, designed to learn contextual representations of DNA. Much like how language models capture grammar and semantics, the Nucleotide Transformer captures patterns in nucleotide sequences (A, C, G, T) that are relevant for downstream biological tasks.

In particular, the model **InstaDeepAI/nucleotide-transformer-v2-100m-multi-species** is a 100M-parameter variant trained across multiple species. It provides embeddings that generalize well across organisms, making it suitable for tasks such as species classification, functional genomics, and evolutionary studies. By fine-tuning this model on insect DNA sequences, it is possible to extract biologically meaningful features that complement image-based representations.

3 Dataset

3.1 Overview

The dataset used in this project consists of paired **DNA sequences** and **images** of insect species. It contains a total of 32,424 records, split into training, validation, and test sets. Each record includes an image of the insect and a corresponding DNA sequence. This multimodal design allows learning from both morphological (image) and genetic (DNA) information.

3.2 Dataset Split

The dataset was divided into training, validation (seen/unseen), and test(seen/unseen) sets as shown in the table below.

Split	Number of Records
Training	13,039
Validation (seen)	3,234
Validation (unseen)	3,721
Test (seen)	4,990
Test (unseen)	7,440
Total	32,424

3.3 Image Data

The image data consists of color images resized to a fixed resolution of 64×64 pixels with 3 channels (RGB). Thus, each image has a shape of $(3, 64, 64)$. These images capture the external morphology of insects, such as body shape, wing patterns, and color distribution.

3.4 DNA Data

The DNA sequences were preprocessed and padded to a maximum sequence length of 1551 nucleotides, ensuring consistent input size for the transformer encoder. Table 1 summarizes the statistics of the DNA strings.

Table 1: DNA sequence statistics after preprocessing.

Property	Value
Maximum length	1551
Minimum length	511
Mean length	~ 660
Padded input length	1551

3.5 Classes

The classification task involves two taxonomic levels: genus and species. Table 2 summarizes the number of classes.

Table 2: Number of classes in the dataset.

Category	Number of Classes
Species	1050
Genus	372

4 Methodology

4.1 Overview

The proposed method follows a multimodal and hierarchical approach for insect classification. Each sample consists of an image and a DNA sequence. We process these inputs separately using two pretrained transformer-based encoders: the **Nucleotide Transformer** for DNA sequences and the **Vision Transformer (ViT)** for images. The resulting embeddings, together with DNA sequence length information, are then combined through an **attention-based fusion module** that produces a unified representation.

On top of this fused representation, the model performs classification in two stages. First, a **genus classifier** predicts the genus of the insect. Then, using the genus–species mapping described earlier, a **species classifier** predicts the species within the identified genus. This hierarchical design reduces complexity compared to direct species-level classification and reflects the natural structure of biological taxonomy.

The entire model is trained end-to-end, with frozen encoders and trainable fusion and classification layers. Losses from genus prediction, species prediction, and overall classification are combined to optimize the model.

4.2 Data Preprocessing

4.2.1 Genus–Species Mapping

We constructed a mapping between each genus and the set of species that belong to it. This mapping reflects the hierarchical nature of taxonomy and is later used to restrict species classification to the set of species within the predicted genus. The largest genus in the dataset contains 23 species, which defines the maximum output size for the species classifier.

Figure 1: Examples of insect images grouped by genus. Each row represents a genus, and the images in that row correspond to different species within the same genus. This illustrates the hierarchical structure of the dataset, where multiple species are nested under a single genus.

4.2.2 DNA Sequences

DNA sequences in the dataset were first stripped of leading and trailing whitespace. We then calculated the length of each DNA string and used it as an additional feature for our model, encoding sequence length into discrete tokens. Finally, the DNA sequences were tokenized and padded to a fixed length, ensuring that all inputs have a uniform size suitable for the transformer encoder.

4.2.3 DNA Length Tokenization

In addition to DNA embeddings, we use sequence length as a discrete feature in our work. The motivation is that DNA sequence length can carry biological information relevant to species discrimination. Instead of treating length as a continuous scalar, we discretize it into categorical tokens that can be embedded and integrated within the fusion module.

Mapping sequence lengths to tokens. We first computed the length of each DNA string after stripping whitespace:

$$\texttt{all\_dna\_len} = \{\ |s| : s \in \texttt{all\_string\_dnas}\ \}.$$

A mapping dictionary `dna_str_len_mapping` was then constructed, where each unique length value is assigned a unique integer ID. The integer token is computed using:

$$\begin{cases} \texttt{strlen\_mapping}[L], & L \in \texttt{str\_len\_mapping} \\ \texttt{len(strlen\_mapping)}, & \text{otherwise.} \end{cases}$$

Tokenization. Applying this function to all DNA lengths yields a tokenized sequence-length array:

$$\{\ \texttt{strlen\_to\_int}(L) : L \in \texttt{all\_dna\_len}\ \}.$$

These tokens are stored as integer IDs and later embedded into dense vectors by a learnable `nn.Embedding` layer within the fusion module.

4.2.4 Images

The insect images were provided in the format $(3, 64, 64)$, corresponding to $(\text{channels}, \text{height}, \text{width})$. I permuted the dimensions to $(64, 64, 3)$, corresponding to $(\text{height}, \text{width}, \text{channels})$. No additional preprocessing such as normalization or augmentation was applied.

d

4.3 DNA Encoder

For processing DNA strings, we adopt the pretrained `nucleotide-transformer-v2-100m-multi-species` model, a transformer-based encoder-only trained on diverse genomic data. We fine-tuned this model on insect species labels using the Hugging Face `Trainer` framework. Fine-tuning was cast as a sequence classification task: a linear classification head was attached to the encoder, with the number of output classes set to the number of species present in the training set. The head was initialized with random weights, while the encoder was initialized from the public checkpoint. After fine-tuning, we discarded the classification head and retained the encoder as a frozen feature extractor for the multimodal model.

Tokenization and Input Representation. Sequences are padded to a fixed length of 1600 tokens, and an attention mask was generated in parallel. Thus, each input sequence s is mapped to a pair $(\mathbf{x}, \mathbf{m})$, where $\mathbf{x} \in \{0, 1, \ldots, V\}^{1600}$ are input IDs and $\mathbf{m} \in \{0, 1\}^{1600}$ is the attention mask.

Fine-Tuning. We trained the model for 10 epochs with the **AdamW** optimizer, using a **learning rate of** 2×10^{-5}, **weight decay of** 0.01. We performed an evaluation every 100 steps, saving checkpoints every 500 steps. The best checkpoint was selected based on validation loss. Accuracy was used as the primary evaluation metric.

Embedding Extraction. Having the fine-tuned model, we drop the classification head, to retrieve the embeddings of the DNA sequence. The latent size by default is set to **512**. We use the embedding of the [CLS] token representing the whole input sequence.

Evaluation. To validate the encoder, we evaluated the fine-tuned model both on the *seen* validation split and on the *unseen* test split. These evaluations provide a measure of how well the nucleotide encoder generalizes before integration into the multimodal pipeline. The results are reported in the next section.

4.4 Image Encoder

For visual features, we employ a Vision Transformer backbone, `google/vit-base-patch16-224`. We fine-tune a classification head on top of the pretrained ViT using the Hugging Face `Trainer`. After fine-tuning, we discard the classifier and retain the base encoder to extract image embeddings for multimodal fusion.

Preprocessing and Tokenization. Raw images are provided in CHW format $(3, 64, 64)$. For dataset construction, each sample is permuted to HWC. Images are then processed by `AutoImageProcessor` associated with the ViT checkpoint, which produces `pixel_values` tensors suitable for the model forward pass.

Fine-Tuning Setup. The number of classes is inferred as $\max(\text{labels}) + 1$. We fine-tune the classifier with:

- batch size 32 (train and eval),
- 10 epochs,
- learning rate 1×10^{-4},
- evaluation every 500 steps and checkpointing every 500 steps,
- selection of the best checkpoint by validation loss (`load_best_model_at_end=True`, `metric_for_best_model=eval_loss`).

Accuracy is tracked as the main metric during evaluation.

Embedding Extraction. For downstream fusion, we again load only the base model, which produces embeddings. The latent size for the image encoder is by default 768. We use the embedding of the [CLS] token representing the whole input sequence.

4.5 Embeddings Fusion

We fuse dna and image embeddings with a lightweight attention–gating block, that ingests three inputs per sample: (i) the discretized DNA-length token t_ℓ, (ii) the DNA embedding $\mathbf{h}_{\text{DNA}} \in R^{d_{\text{dna}}}$, and (iii) the image embedding $\mathbf{z}_{\text{img}} \in R^{d_{\text{img}}}$. The DNA-length token is first embedded via a learnable lookup table $\mathbf{E}_\ell \in R^{n_\ell \times d_\ell}$ (`nn.Embedding`) to produce $\mathbf{e}_\ell \in R^{d_\ell}$. Optional linear projections align the DNA and image channels to new dimensions (`proj_dna_dim`, `proj_img_dim`). An attention value is computed for each of the three embeddings and reweights each vector before concatenation. Finally, A small feed-forward layer maps the concatenated vector to the fused space.

Inputs and embeddings. Let d_ℓ denote the DNA-length embedding size (default 16), and n_ℓ the number of distinct length bins (default 120). We compute

$$\tilde{\mathbf{h}}_{\text{DNA}} = \begin{cases} \mathbf{W}_{\text{dna}}\mathbf{h}_{\text{DNA}}, & \text{if projection enabled} \\ \mathbf{h}_{\text{DNA}}, & \text{otherwise} \end{cases}$$

$$\tilde{\mathbf{z}}_{\text{img}} = \begin{cases} \mathbf{W}_{\text{img}}\mathbf{z}_{\text{img}}, & \text{if projection enabled} \\ \mathbf{z}_{\text{img}}, & \text{otherwise} \end{cases}.$$

During training, a dropout is applied to the projected streams.

Scalar attention gating. Three single-output linear layers (`nn.Linear(·, 1, bias=False)`) produce scalar gates. We avoided transformer-style multi-head attention to make the model less complex and keep the number of parameters small.

$$\alpha_\ell = \mathbf{w}_\ell^\top \mathbf{e}_\ell, \quad \alpha_{\text{dna}} = \mathbf{w}_{\text{dna}}^\top \mathbf{h}_{\text{DNA}}, \quad \alpha_{\text{img}} = \mathbf{w}_{\text{img}}^\top \mathbf{z}_{\text{img}},$$

which rescale each vector:

$$\hat{\mathbf{e}}_\ell = \alpha_\ell\, \mathbf{e}_\ell, \quad \hat{\mathbf{h}}_{\text{DNA}} = \alpha_{\text{dna}}\, \tilde{\mathbf{h}}_{\text{DNA}}, \quad \hat{\mathbf{z}}_{\text{img}} = \alpha_{\text{img}}\, \tilde{\mathbf{z}}_{\text{img}}.$$

The three gated components are concatenated

$$\mathbf{s} = \left[\hat{\mathbf{e}}_\ell \,;\, \hat{\mathbf{h}}_{\text{DNA}} \,;\, \hat{\mathbf{z}}_{\text{img}}\right] \in R^{d_\ell + d'_{\text{dna}} + d'_{\text{img}}},$$

where $d'_{\text{dna}} =$ `proj_dna_dim` (or d_{dna}) and $d'_{\text{img}} =$ `proj_img_dim` (or d_{img}). A final linear layer (`ffn`) maps $\mathbf{s}$ to the fused space $R^{d_{\text{fuse}}}$:

$$\mathbf{f} = \mathbf{W}_{\text{ffn}}\mathbf{s} \in R^{d_{\text{fuse}}},$$

$$d_{\text{fuse}} = \begin{cases} \texttt{fused\_dim}, & \text{if set} \\ d_\ell + d'_{\text{dna}} + d'_{\text{img}}, & \text{otherwise.} \end{cases}$$

The module outputs $(\mathbf{f}, \mathbf{e}_\ell)$, where $\mathbf{f}$ is the fused representation and $\mathbf{e}_\ell$ (the DNA-length embedding) is propagated to downstream heads.

4.6 Hierarchical Classification

We adopt a two-stage hierarchy that mirrors taxonomic structure. Given the fused embedding $\mathbf{f} \in R^{d_{\text{fuse}}}$ (from the Fusion Module) and the DNA-length embedding $\mathbf{e}_\ell \in R^{d_\ell}$, the network first predicts a genus distribution and then produces *genus-conditioned* local species scores. The two outputs are combined into global species logits that span all species in the dataset.

4.6.1 Genus Classification

The genus head receives the concatenation of the DNA-length embedding and the fused embedding,

$$\mathbf{u} = [\,\mathbf{e}_\ell;\, \mathbf{f}\,] \in R^{d_\ell + d_{\text{fuse}}}.$$

A shallow MLP maps $\mathbf{u}$ to genus logits, followed by a softmax over G genera,

$$\mathbf{g} = \text{softmax}\big(\text{MLP}_{\text{genus}}(\mathbf{u})\big) \in \Delta^{G-1}.$$

Concretely, the head consists of $\texttt{Linear}(d_\ell + d_{\text{fuse}} \to h)$, `Sigmoid`, `Dropout`, and $\texttt{Linear}(h \to G)$ layers. During training, we optionally apply *teacher forcing* by replacing $\mathbf{g}$ with the one-hot encoding of the ground-truth genus when available:

$$\mathbf{g} \leftarrow \text{one\_hot}(y_{\text{genus}}, G).$$

4.6.2 Genus–Species Classification

To predict species within each genus, we form a reduced representation of the fused embedding and a dense embedding of the genus distribution:

$$\tilde{\mathbf{f}} = \text{Dropout}\big(\mathbf{W}_{\text{red}}\mathbf{f}\big) \in R^{d_{\text{red}}},$$

$$\mathbf{e}_g = \text{Dropout}\big(\mathbf{W}_g\mathbf{g}\big) \in R^{d_g}.$$

These are concatenated with the DNA-length embedding and fed to a local-species decoder:

$$\mathbf{v} = [\,\mathbf{e}_\ell;\, \mathbf{e}_g;\, \tilde{\mathbf{f}}\,], \qquad \mathbf{s} = \text{MLP}_{\text{local}}(\mathbf{v}) \in R^{S_{\max}},$$

where $S_{\max}$ is the maximum number of species in any genus. The decoder is a $\texttt{Linear}(d_\ell + d_g + d_{\text{red}} \to$

h_s), `Sigmoid`, `Dropout`, `Linear`$(h_s \to S_{\max})$ stack. Interpreting $\mathbf{s}$ *per genus* gives a matrix of local species scores of shape $[B, G, S_{\max}]$ when broadcast across genera.

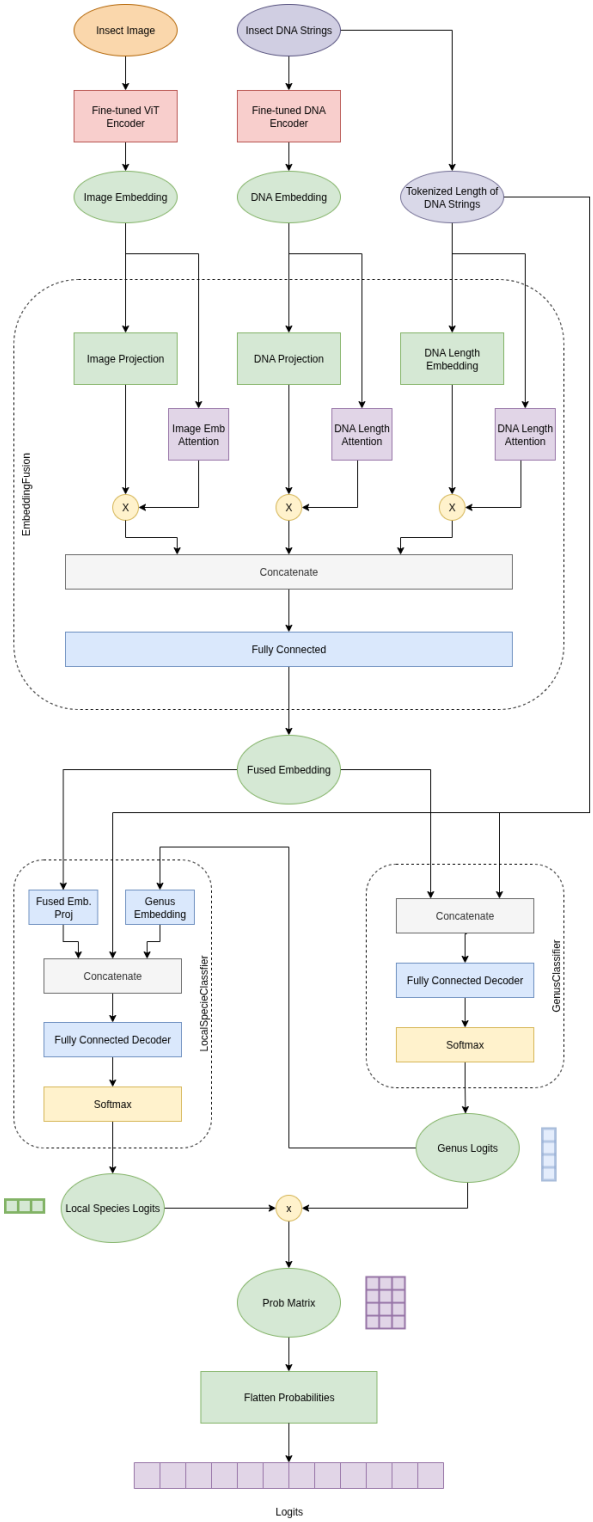

Figure 2: Overall methodology: DNA and image encoders produce embeddings that are fused. The fused embedding is passed through a hierarchical classification pipeline, which first predicts the genus and then the species within that genus.

4.6.3 Final Classification

Global species probabilities are obtained by combining genus probabilities with the genus-conditional local species scores. Let $\mathbf{g} \in R^{B\times G}$ and $\mathbf{s} \in R^{B\times S_{\max}}$. We first form an outer product (broadcasting $\mathbf{s}$ over G) to create a *probability matrix*

$$\mathbf{P} = \mathbf{g} \otimes \mathbf{s} \in R^{B\times G\times S_{\max}}, \quad \mathbf{P}_{b,g,s} = \mathbf{g}_{b,g}\, \mathbf{s}_{b,s}.$$

Because different genera have different numbers of species, we then *flatten* $\mathbf{P}$ into global species logits $\mathbf{z} \in R^{B\times S}$ by scattering the first $|\mathcal{S}_g|$ entries along the slice corresponding to genus g into the global indices of species set $\mathcal{S}_g$:

$$\mathbf{z}_{b,\mathcal{S}_g} \leftarrow \mathbf{P}_{b,g,\,0:|\mathcal{S}_g|}, \qquad \forall g \in \{1,\dots,G\}.$$

This yields a single vector over all S species, consistent with the dataset's genus→species mapping.

Training objective. The total loss is a weighted sum of cross-entropies:

$$\mathcal{L} = \alpha\ \mathrm{CE}(\mathbf{z},\, y_{\text{species}}) + \beta\ \mathrm{CE}(\mathbf{g},\, y_{\text{genus}})$$

where y_{species} is the global species label, and y_{genus} is the genus label. The coefficients (α, β) control the trade-off among specie and genus classification.

4.7 Experimental Setup

At the start of this work, DNABert [5] was used as the DNA encoder. However, DNABert (V2 and later) supports sequences up to 512 nucleotides, which was too short for our dataset. Therefore, we switched to the Nucleotide Transformer [4], which allows a maximum length of 2048 nucleotides and is better suited for our task. In this section, we report the setup of the best-performing model, after testing several experimental configurations.

4.7.1 Evaluation Metric

We used **accuracy** as the main evaluation metric, computed on the validation split.

4.7.2 Hardware setup:

All experiments were run on the university cluster using **4 NVIDIA L40 GPUs** in parallel. Because the Nucleotide Transformer requires processing DNA sequences of length up to 1600, GPU memory (GRAM) was a limiting factor. We therefore used a small batch size for DNA sequence training.

4.7.3 Implementation Details

Vision Transformer (ViT) Fine-tuning. The image encoder was fine-tuned using the following hyperparameters:

DNA Encoder Fine-tuning. The nucleotide encoder was fine-tuned with the following setup:

Table 3: Training setup for the ViT image encoder.

Parameter	Value
Batch size (train/eval)	32 / 32
Epochs	10
Learning rate	1×10^{-4}
Logging steps	50
Evaluation steps	500
Save steps	500
Best model selection	Validation loss

Table 4: Training setup for the DNA encoder.

Parameter	Value
Batch size (train/eval)	4 / 4
Gradient accumulation	4 (effective batch size 16)
Epochs	10
Learning rate	2×10^{-5}
Weight decay	0.01
Logging steps	50
Evaluation steps	100
Save steps	500
Mixed precision	FP16
Best model selection	Validation loss

Multimodal Model Training. When training the full multimodal classifier, we used precomputed DNA and image embeddings. This allowed us to use a larger batch size, since no tokenizer or encoder forward passes were required during training.

Table 5: Training setup for the multimodal classifier.

Parameter	Value
Batch size (train/eval)	512 / 512
Epochs	200
Learning rate	1×10^{-4}
Weight decay	0.01
Evaluation steps	200
Save steps	400
Scheduler	Linear decay
Logging steps	50

4.7.4 Multimodal Model Architecture

Our final model combines DNA embeddings, image embeddings, and DNA sequence length tokens through an attention-based fusion module, followed by hierarchical genus and species classifiers. The configuration of each component is summarized in Table 6.

4.8 Training Curves

Figure 8 shows that the multimodal model converges stably, with steadily decreasing losses and validation accuracy that rises quickly before plateauing at a stable level.

0.32

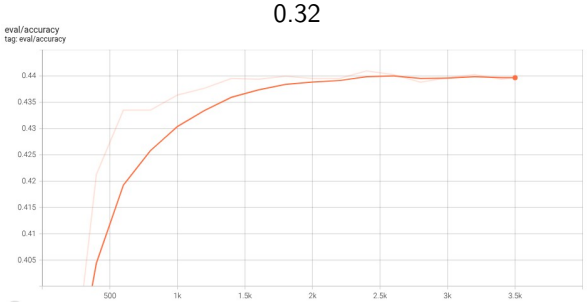

Figure 3: Validation accuracy over training steps.

0.32

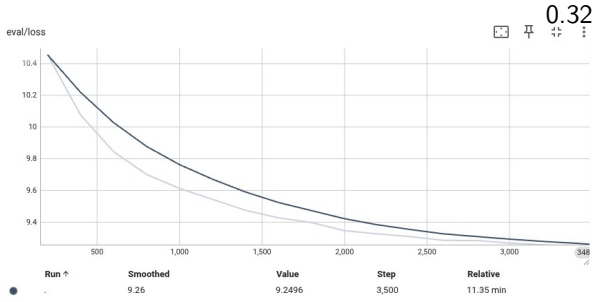

Figure 4: Validation loss over training steps.

0.48

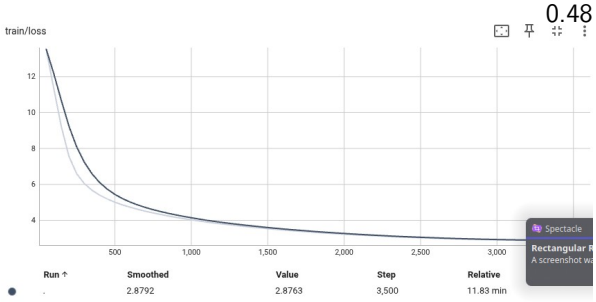

Figure 5: Training loss over steps.

0.32

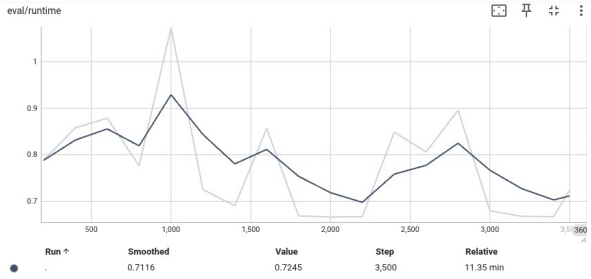

Figure 6: Validation runtime per evaluation.

0.48

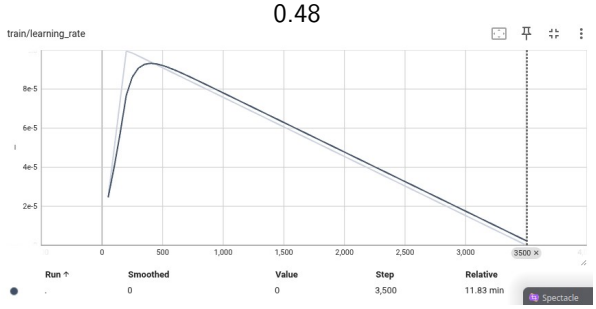

Figure 7: Learning rate schedule (warmup + decay).

Figure 8: Training dynamics of the multimodal model. The plots show evaluation accuracy, evaluation loss, runtime per evaluation, learning rate schedule, and training loss. Accuracy and loss curves confirm convergence, while runtime fluctuations reflect GPU load balancing during evaluation.

Table 6: Configuration of the implemented multimodal architecture.

Module	Configuration
Fusion Embedder	DNA embedding dimension: 512; Image embedding dimension: 768; Projected DNA dim: 96; Projected Image dim: 128; DNA length embedding dim: 32 (120 distinct tokens); Fused dimension: 256; Dropout: 0.2.
Genus Classifier	Input: concatenation of fused embedding (256) + DNA length embedding (32); Hidden dimension: 744; Output: 372 genera; Dropout: 0.2.
Genus-Species Classifier	Reduced fused dim: 128; Genus embedding dim: 64; Specie decoder hidden dim: 256; Max species per genus: 23; DNA length embedding dim: 32; Dropout: 0.2.

This architecture ensures that:

1. The fusion module aligns DNA, image, and length features into a shared 256-dimensional representation.
2. The genus classifier operates at the genus level, producing a probability distribution over 372 genera.
3. The local species classifier leverages the predicted genus context to produce species-level predictions within each genus (up to 23 species).

4.8.1 Baselines

We compare our multimodal model against two unimodal baselines:

1. **Fine-tuned Vision Transformer (ViT)** — using only insect images.
2. **Fine-tuned Nucleotide Transformer** — using only DNA sequences.

4.9 Fine-Tuned Unimodal Models

Both unimodal baselines — the fine-tuned Vision Transformer (ViT) and the Nucleotide Transformer — achieve good performance on **seen species**, but fail to generalize to **unseen species**. The ViT reaches over 85% accuracy on validation seen species and nearly 70% on test seen species, but drops to about 1% on unseen species. Similarly, the DNA encoder achieves 79% validation accuracy and 64% test accuracy on seen species, but completely collapses on unseen species (0% accuracy). These results demonstrate that unimodal approaches essentially memorize the training distribution and are unable to perform zero-shot generalization.

Overall, both models highlight the challenge of species classification under distribution shifts: they perform well on seen taxa but fail to extend to unseen species, which motivates the need for multimodal and hierarchical modeling.

4.10 Multimodal Model Results

The proposed multimodal architecture achieves the best overall genus and species classification performance compared to unimodal baselines. As shown in Table 7, the model reaches **91.71%** validation accuracy and **77.07%** test accuracy on seen species, outperforming both the fine-tuned ViT and DNA encoders. More importantly, while unimodal models fail completely on unseen species (ViT: $\sim 1\%$, DNA: 0%), the multimodal model improves generalization, reaching **7.31%** validation accuracy and **10.19%** test accuracy on unseen species. Although these values remain low in absolute terms, they demonstrate that multimodal fusion enables partial transfer across taxa not observed during training.

Table 7: Performance comparison of unimodal baselines and the multimodal model on seen and unseen species.

Data Split	ViT	DNA Enc.	Multimodal
Val. Seen	85.28%	78.78%	**91.71%**
Val. Unseen	1.37%	0.00%	**7.31%**
Test Seen	69.52%	64.37%	**77.07%**
Test Unseen	1.59%	0.00%	**10.19%**

Table 8: Genus classification accuracy (%) of the multimodal model under different training strategies.

Data Split	Joint (α=2, β=2)	Two-stage
Val. Seen	48.39	60.14
Val. Unseen	49.53	42.54
Test Seen	48.04	57.84
Test Unseen	46.41	37.45

5 Conclusion

A clear and concise conclusion, containing key takeaways of the article or report. It is useful to give the readers a huge summary of what they just read, highlighting the main takeaway from the report. It should provide the key information that the author wishes the reader to remember most when reading the report. Although this sounds very similar to the abstract, this is generally true, but make sure the content of both the abstract and the conclusion do not overlap. Basically,

the main body should have an elaborate discussion while summarised in the conclusion.

References

[1] L. Nanni, M. D. Gobbi, R. D. A. M. Junior, and D. Fusaro. Advancing taxonomy with machine learning: A hybrid ensemble for species and genus classification. *Algorithms*, 18(2):105, 2025. doi: 10.3390/a18020105.

[2] Sarkhan Badirli, Christine Picard, George Mohler, Frannie Richert, Zeynep Akata, and Murat Dundar. Classifying the unknown: Insect identification with deep hierarchical bayesian learning. *Methods in Ecology and Evolution*, 14, 04 2023. doi: 10.1111/2041-210X.14104.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[4] H Dalla-Torre, L Gonzalez, J Mendoza-Revilla, N Lopez Carranza, A. H. Grzywaczewski, F Oteri, C Dallago, E Trop, B. P. de Almeida, H Sirelkhatim, G Richard, M Skwark, K Beguir, M Lopez, and T Pierrot. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22 (2):287–297, February 2025. doi: 10.1038/s41592-024-02523-z. Epub 2024 Nov 28.

[5] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15): 2112–2120, 02 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083.

A Appendix

The full source code is available at: https://github.com/amin-tehrani/insect-classification-deeplearning

Fine-tuned models in Hugging Face: Finetuned ViT: https://huggingface.co/amintehrani/vit-insects-finetuned7

Finetuned nucleotide Transformer: https://huggingface.co/amintehrani/dnaencoder-insects-finetuned