Probabilistic Model for Amur Tiger Re-identification

Obafemi Jinadu **Johnathan Oneal**

1 Introduction

The task is to re-identify tigers in the wild, re-identification is a more nuanced form of classification where instead of just classifying the type of animal (tiger in this case), the model is able to identify the tiger on a "first-name basis". Monitoring the population and migration patterns of endangered species is crucial for wildlife conservation. The conventional approach for this task is animal tagging where humans manually place tags around these animals, this is a tedious process that does not scale to large populations. Leveraging computer vision, machine learning and probabilistic techniques this process can be automated via camera trap footage. The dataset for these type of tasks are often limited and largely imbalanced.

A probabilistic model for this task will be useful as it can help provide a comprehensive understanding of the uncertainty associated with predictions. Accounting for uncertainty in the re-identification task is important because oftentimes there is severely limited and largely imbalanced data to train models on. In this project, for example, the model has to learn to classify 107 tiger entities on just 1,887 training samples (an average of 17 training images per tiger) also, these animals tend to have features that are hard to distinguish. Thus, it is crucial to know how confident our model is about the predictions it generates. Uncertainty tells us the "confidence" in our model's predictions. Also, with adequate probabilistic modeling, we can tell if we are over-committing to parameters in the training process.

2 Data and Analysis Plan

The dataset is called Amur Tiger Re-identification in the Wild (ATRW) [1]. Created for two tasks i) re-identification and ii) keypoint-based pose estimation of 92 individual amur tigers. Each observation is an RGB tiger image array of size PxQx3 (each image is of a different resolution). In total, the dataset consists of 5,160 images, where the train set has 3,394 images, and the test set has 1,766 images. However, for the specific task of interest, re-identification, the sub-dataset for training/validation containing 1,887 images with corresponding annotations are publicly available. For the test subset, the annotations are withheld by the organizers. The dataset annotations come as CSV files with ground truth information i.e., a 2-column file where the first column corresponds to the tiger name/id (which is just an integer value), and the second column corresponds to the image file. The dataset is largely imbalanced; some tigers have as high as 98 images while some tigers have as low as 10 images to train on this can be seen in Fig. 1. To address this, some data augmentation was carried out to create balance. The augmentation involved generating synthetic images for each tiger by horizontally flipping, and rotating the images at a range of 20^o. This was done to provide 100 images per class. So for the tiger with 98 images, 2 synthetic images were generated, and for the tiger with 10 images 90 images were synthesized. Also, since each image has a different resolution, all images are resized to a consistent resolution of 200 by 300 by 3. These steps are carried out prior to training on the neural network.

The dataset contains 92 individual tigers, the authors of the dataset state that the tiger stripe pattern is the most informative marker of the tiger. However, the left and right sides of the Amur tigers have different stripe patterns (shown in Fig. 2), and it is quite rare to capture both sides of the tiger in the wild environment therefore each side of the same tiger is treated as a different entity due to this, the dataset contains 107 unique entities to be classified rather than the actual 92 individual amur

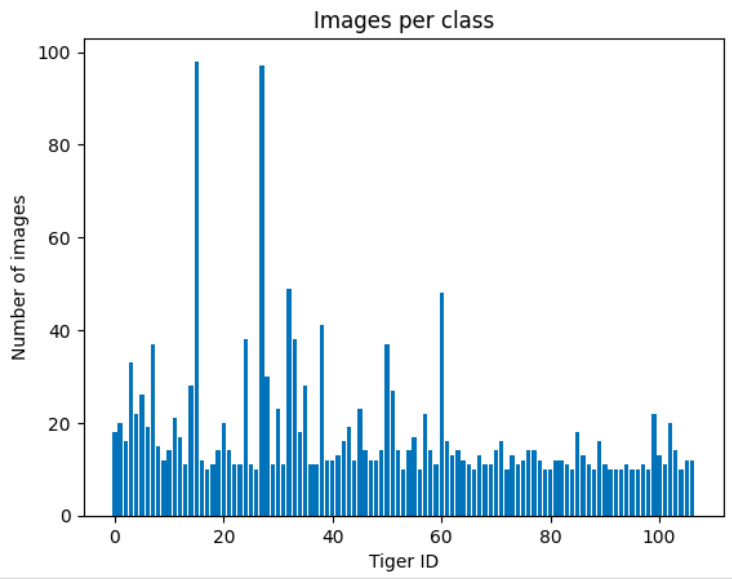
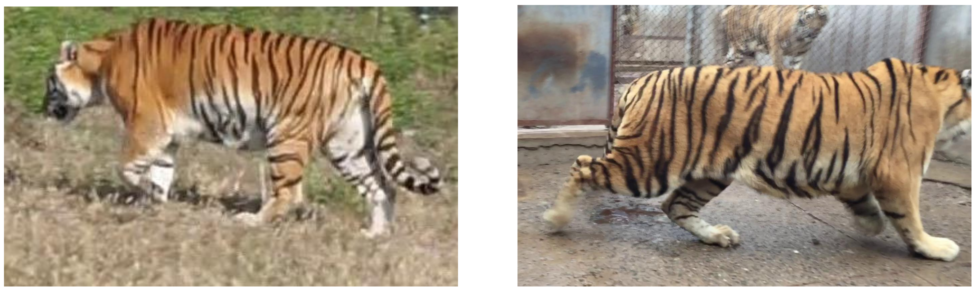


Figure 1: Dataset Distribution showing the number of images of each tiger in the trainset: shows class imbalance)



(a) Tiger left side

(b) Tiger right side

Figure 2: Left and Right Side of Tiger with Different Stripe Pattern [1]

tigers. The performance metric used is the categorical cross-entropy or softmax loss. This is ideal for a multi-class classification task which is exactly what re-identification is. Cross-entropy loss quantifies the accuracy of a classifier model by penalizing misclassifications. Along with the softmax loss, we also consider accuracy (which quantifies the correctness of the model's predictions i.e. how close the predictions are to the true labels). The test set images provided do not have corresponding annotations. Therefore cannot be used for our analysis as there is no way to compute the loss and accuracy of predictions without ground truth. Local test and validation datasets were created using a train-test split. Specifically, the original training set is split into 80% and 20%. The 80% (approximately 1,509 images) were used for training. While the 20% was further divided into equal halves one half corresponds to the validation set and the other half is our local test set, with test and validation consisting of 189 images each. These splits are carried out in a stratified fashion to keep the ratio of labels consistent across all data subsets. We believe this split works for this task because using 80% of the initial training set for training is not too small to hurt the performance and the 20% for the validation and testing set is also sufficient unseen data for the model to validate and test on.

3 Baseline Method

The probabilistic model considered is the likelihood and the baseline method would be maximizing the likelihood. The maximum likelihood estimation is equivalent to obtaining the model weights that minimize the categorical cross-entropy loss of our neural network model. Formally:

- Given N observations input/output pairs: $\{x_n, t_n\}_{n=1}^{N}$ where x_n is a 200x300x3 RGB re-sized image and t_n is the corresponding id of the tiger (output)
- $\phi(x_n)$: features in the last layer of neural network where $\phi(x_n)$ is a vector of size 256. It can be thought of as the basis function an input image x_n has been transformed to after passing through previous layers of the neural network.
- $t_n \in \{1, 2, 3...C\}$ where C is the count of tiger entities i.e. C = 107.
- w or $w_{1:C}$: a 256 x 107 matrix corresponding to the weights of the neural network's last layer.
- probability that a tiger x_n belongs to class $t_n = c$ given weights, w is given by the softmax function below:

$$p(t_n = c|x_n, w) = \frac{e^{w_c^T \phi(x_n)}}{\sum_{k=1}^{C} e^{w_k^T \phi(x_n)}} = y_c(\phi) \tag{1}$$

We define the likelihood, making the (iid) assumption i.e. independent and identically distributed:

$$p(\{t_n\}_{n=1}^{N} | \{x_n\}_{n=1}^{N}, w) = \prod_{n=1}^{N} p(t_n = c|x_n, w) \tag{2}$$

Using one-hot encoding scheme where the target t_n for a feature $\phi(x_n)$ belonging to class $\mathcal{C}_c$ is a binary vector of size 107 and all elements are zero except for element c which equals one. Adapted from Eq 4.107 of Bishop textbook the likelihood function becomes:

$$p(\{t_n\}_{n=1}^{N} | \{x_n\}_{n=1}^{N}, w) = \prod_{n=1}^{N} \prod_{c=1}^{C} y_{nc}^{t_{nc}}, \text{ where } y_{nc} = y_c(\phi_n) \tag{3}$$

We wish to maximize the likelihood i.e. $\arg\max_w p(\{t_n\}_{n=1}^{N} | \{x_n\}_{n=1}^{N}, w)$ by convention, this is equivalent to minimizing the negative of the likelihood, and for numerical stability, we take the log:

$$E(w_{1:C}) = -log_e p(t = \{t_n\}_{n=1}^{N} | w) = -\sum_{n=1}^{N} \sum_{c=1}^{C} t_{nc} log_e y_{nc} \tag{4}$$

Therefore, the Maximum Log Likilihood corresponds to $\arg\min_w E(w_{1:C})$. Where $E(w_{1:C})$ is exactly the categorical cross entropy loss. To achieve this maximum likelihood weight, we use the iterative first-order gradient descent algorithm where each update is given by:

$$w_{new} = w_{old} - \epsilon g \tag{5}$$

where ϵ is the learning rate (which is our tunable hyper-parameter) and gradient, g is given by:

$$g = \nabla_{w_j} E(w_{1:C}) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \phi(x_n) \tag{6}$$

Specifically, we use the stochastic gradient descent (SGD) algorithm where weight updates are performed on a single example at a time. The EfficientNetB3[2] neural network with imageNet [3] weights are used as the base model to train the dataset and all the base model layers are frozen and training to obtain the weights that maximize the likelihood is carried out on the final layers. The model is trained with a batch size of 32 for 10 epochs, where the first 7 epochs use a learning rate of 0.01 and for the last 3 epochs the learning rate is reduced to 0.0009. The software package used is TensorFlow [4] and CUDA for GPU hardware access on the Google Colab Pro environment.

4 Upgrade Method

Maximum likelihood estimation (our baseline method) is known to severely overfit given limited training data such as this, where we have 1,887 training data on 107 tiger classes, with some tigers having as low as 10 images to train on. To address this overfitting challenge on our baseline method, we explore the Maximum A Posteriori (MAP) estimation. Specifically, we include a prior distribution $p(w)$ to capture assumptions about weights w before observing the data. Accounting for this prior distribution along with the likelihood $p(\{t_n\}_{n=1}^{N}|\{x_n\}_{n=1}^{N}, w)$ allows us to evaluate the uncertainty in weights w after observing the dataset in the form of a posterior probability $p(w|\{x_n, t_n\}_{n=1}^{N})$. From Baye's rule:

$$\text{posterior} \propto \text{likelihood x prior} \tag{7}$$

$$p(w|\{x_n, t_n\}_{n=1}^{N}) = \frac{p(\{t_n\}_{n=1}^{N}|\{x_n\}_{n=1}^{N}, w)p(w)}{p(\{x_n\}_{n=1}^{N})} \tag{8}$$

where the denominator $p(\{x_n\}_{n=1}^{N})$ is called the marginal evidence, and is a normalization constant which ensures that the posterior distribution is a valid probability density. It is constant with respect to the weights, and can simply be ignored in our analysis:

$$\text{posterior} \approx \text{likelihood x prior} \tag{9}$$

Specifically, the prior distribution assumption considered is a zero-mean isotropic Multivariate Gaussian distribution defined by:

$$p(w) = \mathcal{N}(w|0_D, \alpha^{-1}I_D) \tag{10}$$

where $\alpha > 0$ is the prior precision parameter, since model weight, w is a 256 x 107 matrix it is flattened to a D size vector, $0_D \in \mathbb{R}^{\mathbb{D}}$ is an all-zero vector, I_D is the D x D identity matrix . The multivariate zero-mean isotropic gaussian prior's PDF reduces to:

$$p(w) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\alpha^{-1}I|^{1/2}} exp(-\frac{1}{2\alpha^{-1}} w^T w) \tag{11}$$

In log-space, the prior is:

$$log_e p(w) = -\frac{D}{2} log_e(2\pi) - \frac{1}{2} log_e|\alpha^{-1}I| + (\frac{1}{2\sigma^2} w^T w) \tag{12}$$

The first two terms are constants with respect to weights and can be ignored for our analysis:

$$log_e p(w) = -\frac{\alpha}{2} w^T w \tag{13}$$

The posterior in log-space is:

$$log_e p(w|\{x_n, t_n\}_{n=1}^{N}) = log_e p(\{t_n\}_{n=1}^{N}|\{x_n\}_{n=1}^{N}, w) + log_e p(w) \tag{14}$$

Here our new goal is to obtain the model weights, W_{MAP} that maximizes the posterior distribution i.e. the Maximum a Posteriori (MAP) estimation which is simply $\arg\max_w p(w_{1:C}|\{x_n, t_n\}_{n=1}^{N})$ again, this is equivalent to minimizing the negative of the posterior (in log space for numerical stability) i.e:

$$W_{MAP} = \arg\min_w -log_e p(w|\{x_n, t_n\}_{n=1}^{N}) \tag{15}$$

This is simplify a modification of our baseline categorical cross entropy loss/ negative log likelihood $E(w_{1:C})$ with the prior included:

$$W_{MAP} = \arg\min_w E(w_{1:C}) - log_e p(w) \tag{16}$$

$$W_{MAP} = \arg\min_w - \sum_{n=1}^{N} \sum_{c=1}^{C} t_{nk} log_e y_{nc} + \frac{\alpha}{2} w^T w \tag{17}$$

Observing eqn. 17, we can interpret that the zero-mean, isotropic multivariate gaussian prior on the network weights simply reduces to an L2 norm on the parameter weights, giving an L2 regularized categorical cross-entropy loss. When $\alpha = 0$, we obtain our baseline.

Similarly, as in our baseline case, to achieve the model weights that maximizes the posterior (W_{MAP}), we use the iterative first-order gradient descent algorithm where each update is given by eqn. 5 and gradient g in this case is:

$$g = \nabla_{w_j}[E(w_{1:C}) + \frac{\alpha}{2}w^T w] = \sum_{n=1}^{N}(y_{nj} - t_{nj})\phi(x_n) + \alpha w_j \tag{18}$$

the weight update eqn. 5 becomes:

$$w_{new} = w_{old} - \epsilon(\sum_{n=1}^{N}(y_{nj} - t_{nj})\phi(x_n) + \alpha w_j) \tag{19}$$

where ϵ and α the learning rate and prior hyper-parameter respectively and are both tunable hyper-parameters for our model. The stochastic gradient descent (SGD) algorithm is also maintained here. The EfficientNetB3[2] neural network with ImageNet[3] weights is also maintained to train the dataset and all the base model layers are frozen, and training to obtain W_{MAP} is performed on the final layers. Similarly, the model is trained with a batch size of 32 for 10 epochs, where the first 7 epochs use a learning rate of 0.01, and for the last 3 epochs, the learning rate is reduced to 0.0009 (these learning rates were experimentally determined to give the best performance on both the baseline and upgrade). The key difference between our upgrade implementation and our baseline is that our modified loss function i.e., the L2 regularized categorical cross-entropy loss function where the prior precision α is tuned with several values experimented and 0.002 gives the best results as penalty values larger than 0.002 tend to overwhelm the categorical cross-entropy loss thus reducing performance, and values lower than 0.002 tend to reduce the effect of the penalty effectively giving us our baseline results. The software packages used were TensorFlow [4] and CUDA for GPU access on the Google Colab environment, where the pro version of Google Colab was used for access to more compute units.

5 Results

To show the hypothetical flaw of our baseline (MLE) i.e. overfitting and poorer performance with limited data, and how our proposed upgrade (MAP) fairs in such scenarios. We perform carry out our experiments in two phases:

1. Baseline and upgrade implementation without the data augmentation pre-processing step (here we have 1,509 images to train on).
2. Baseline and upgrade implementation with data augmentation pre-processing step (here, images are synthesized so that every class/tiger has exactly 100 training images, giving a total of 10,700 to images to train on).

For reproducible results, we set seeds consistently both globally and locally. However, CUDA has non-deterministic backend random initializations that cannot the altered. In an ideal scenario, we would perform several iterations and get the average. However, due to limited compute resources and the time it takes train a model (it takes about an hour to train a model with GPU and about 12 hours to train on TPU provided by Google Colab). This makes having several runs in-feasible. We did run few iterations giving very consistent results. The results of our baseline and upgrade on both experimental phases considered are given in table 1 and Fig. 3 below:

Table 1: Baseline and Upgrade Result Comparison with and without data augmentation on training, validation and test data subsets

Experiment	Train Loss	Train Acc.	Val. Loss	Val. Acc.	Test Loss	Test Acc.
Baseline	2.78	36.47%	2.67	43.39	2.72	51.32%
Upgrade	2.38	44.23%	2.34	51.32	2.22	57.67%
Baseline + Augmentation	1.04	72.66%	1.03	74.07	1.00	77.78%
Upgrade + Augmentation	1.05	72.99%	1.10	78.84	1.07	79.89%

We observed that without the data augmentation pre-processing step that synthesizes more data and balances the tiger classes, the MAP estimation upgrade outperformed the baseline ML model on all metrics considered (loss and accuracy) as expected by about 8% and 6% on the validation and test set respectively in accuracy as captured in table 1 and Fig. 3. This aligned with our hypothesis as the tiger data without augmentation has a significantly limited number of images per class with a large class imbalance, thus the MAP model helped regularize the estimation and reduced the variance of the estimate(reduced over-fitting).

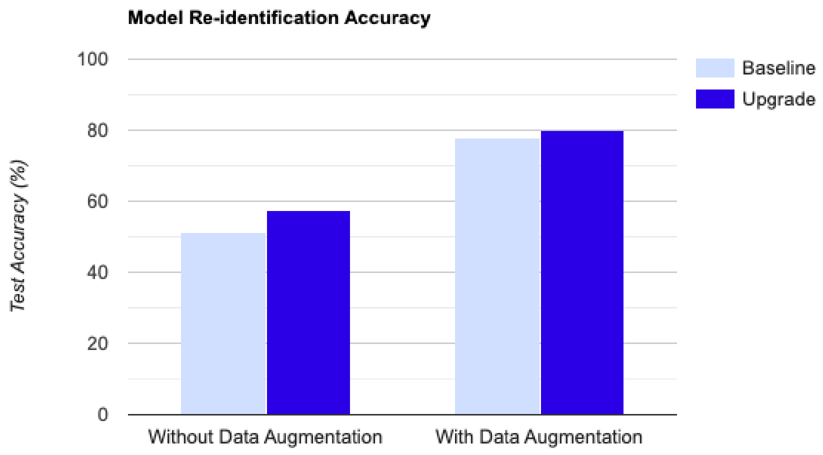


Figure 3: Baseline vs Upgrade accuracy with and without Data Augmentation

With data augmentation, we again observe that the upgrade outperformed the baseline model in accuracy however by a smaller margin of 4% and 2% in the validation and test accuracy. In terms of loss, both the baseline and upgrade perform comparably. The upgrade allows for more reliable incorporation of uncertainty in the parameter estimates with limited data, which can be helpful in cases where there is variability in the stripe patterns due to factors such as illumination conditions, camera angles, or changes in the tiger's coat over time. However, as we augment the dataset, we have more training samples and have essentially eliminated the class imbalance issue, and the advantages of incorporating the prior distribution for a MAP estimation start to slightly diminish.

References

[1] Li, S., Li, J., Tang, H., Qian, R. and Lin, W., 2019. ATRW: a benchmark for Amur tiger re-identification in the wild. arXiv preprint arXiv:1906.05586.

[2] Tan, M. and Le, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

[3] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[4] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.