Supplementary Material for Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup

Jang-Hyun Kim [1 2] **Wonho Choo** [1 2] **Hyun Oh Song** [1 2]

A. Proofs

A.1. Proof of Lemma1

Lemma 1. *If ψ, ϕ satisfy pairwise submodularity and β, $\gamma \in \mathbb{R}_+$, then $\beta\psi + \gamma\phi$ satisfies pairwise submodularity.*

Proof. By the assumption, β, $\gamma \in \mathbb{R}_+$, and by using pairwise submodularity of ψ and ϕ, $\forall x, y \in \mathcal{L}$,

$$\begin{aligned}
&(\beta\psi + \gamma\phi)(x, x) + (\beta\psi + \gamma\phi)(y, y) \\
&= \beta(\psi(x, x) + \psi(y, y)) + \gamma(\phi(x, x) + \phi(y, y)) \\
&\leq \beta(\psi(x, y) + \psi(y, x)) + \gamma(\phi(x, y) + \phi(y, x)) \\
&= (\beta\psi + \gamma\phi)(x, y) + (\beta\psi + \gamma\phi)(y, x).
\end{aligned}$$

□

A.2. Proof of Proposition 5

Proposition 5. *Algorithm 1 converges to a local-minimum with respect to the update rule at most $n(n-1)/2+1$ steps.*

Proof. Let $C^{(0)}$ has a minimum at (i_1, j_1). Then, $\forall t = 0, 1, ...,$ $\Pi^{(t)}_{win}[i_1, j_1] = 1$. Next, let's define $I_2 = \{(i, j)|i \neq i_1, j \neq j_1\}$ and $(i_2, j_2) = \text{argmin}_{(i,j)\in I_2} C^{(1)}[i, j]$. If $\Pi^{(0)}[i_2, j_1] = 1$, $C^{(0)}[i_2, j_1]$ will be added by the large value, and thus, $\Pi^{(1)}_{win}[i_2, j_2] = 1$. Otherwise, if $\Pi^{(0)}[i_2, j_1] = 0$, then $C^{(0)}[i_2, j_2] \leq C^{(0)}[i_2, j_1]$ and $\Pi^{(0)}_{win}[i_2, j_2] = 1$. By the definition of (i_2, j_2), $\forall t \geq 1$, $\Pi^{(t)}_{win}[i_2, j_2] = 1$.
To use induction, let's define $I_k = \{(i, j)|i \notin \{i_1, ..., i_{k-1}\}, j \notin \{j_1, ..., j_{k-1}\}\}$ and let a_{k-1} denote a minimal step number at which $\Pi^{(t)}_{win}[i_{k-1}, j_{k-1}] = 1$, $\forall t \geq a_{k-1}$. Let's define $(i_k, j_k) = \text{argmin}_{(i,j)\in I_k} C^{(a_{k-1}+1)}[i, j]$. If $\exists j \in \{j_1, ..., j_{k-1}\}$, $\Pi^{(a_{k-1})}_{win}[i_k, j] = 1$, then $\forall t \geq a_{k-1} + k - 1$, $\Pi^{(t)}_{win}[i_k, j_k] = 1$. If not, $\Pi^{(a_{k-1})}_{win}[i_k, j_k] = 1$, and $\forall t \geq a_{k-1}$, $\Pi^{(t)}_{win}[i_k, j_k] = 1$. Thus, $a_k \leq a_{k-1} + k - 1$.
Finally, by induction, $a_n \leq n(n-1)/2$ which means there are no more updates of Π_{win} after $n(n-1)/2+1$ steps. □

B. Analysis of Algorithms

B.1. Comparison Experiments for Algorithm 1

Figure 1 and Figure 2 show the comparison results of Algorithm 1 with the exact Hungarian algorithm on 100 random samples per each vertex size n. Note that, the size of a transport plan is $n \times n$. In the simulation, we generate a random cost matrix $C' = C - s(x)z^\intercal$, where $s(x)$ is sampled from a uniform distribution and the mask z is sampled from a bernoulli distribution with probability $p = 0.5$. In the case of $n = 1024$, Algorithm 1 is about 8.6 times faster than the exact algorithm, with relative error of 0.0005. For comparison, we use lapjv solver from library[1] as the exact optimizer, which to the best of our knowledge is the fastest solver.

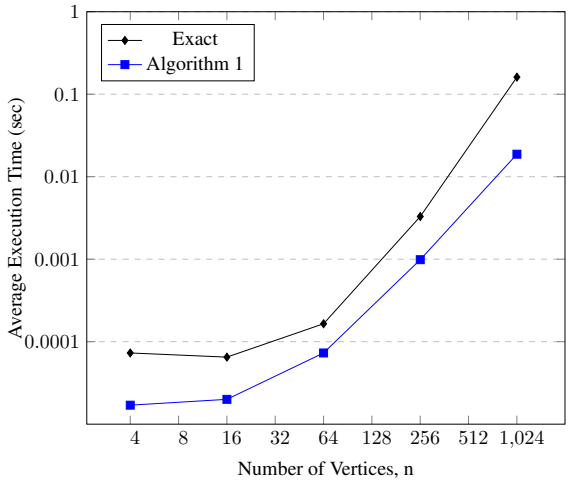

Figure 1. Comparison of average execution time (log-log scale) to solve Equation (5) between the exact solver (black) and Algorithm 1 (blue). Execution times are mean of 100 trials.

[1]Department of Computer Science and Engineering, Seoul National University, Seoul, Korea [2]Neural Processing Research Center. Correspondence to: Hyun Oh Song <hyunoh@snu.ac.kr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]https://github.com/berhane/LAP-solvers

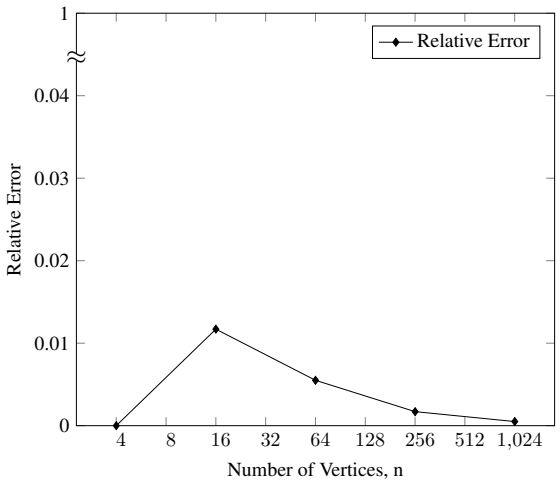

Figure 2. Relative errors of objective function value f between Algorithm 1 (*alg*) and random assignment (*random*). Relative error is calculated as $e_a/(e_a + e_r)$, where $e_a = f_{alg} - f_{exact}, e_r = f_{random} - f_{exact}$.

In addition to the simulation test, we train classifiers with ten different random seeds to compare the Top-1 accuracy between using the Hungarian algorithm and Algorithm 1. In this experiment, we train WRN28-10 on CIFAR100 for 400 epochs. In summary, the mean difference of the Top-1 accuracy is -0.025, with a standard deviation of 0.239. Besides, we perform a two-sided paired t-test to check the statistical insignificance. As a result, T-statistics is -0.105 with a P-value of 0.919, which means there is no evidence for a statistical difference between the two methods.

B.2. Convergence of Alternating Minimization

We solve the optimization problem of a mask and transport plans via alternating minimization for one-cycle for computational efficiency. In this subsection, we analyze the convergence property of the alternating algorithm by optimizing 1,000 CIFAR100 image pairs with various numbers of regions. As a result, we observe that the most of optimal masks of the images are not changed after the first cycle (*i.e.,* comparison between one-cycle and multi-cycle), and hence the optimal transport plans are not changed. For transport cost coefficient ξ in [0.5, 0.8], which shows the best performance, less than 0.2% of the final mixed images change after the first cycle. Also, the ratio of pixels changed after the first cycle is less than 0.001%.

We believe that this result is due to the mutual complement of the optimal mask and the optimal transport. That is, the optimal mask assigns the output regions so that the remained saliency of each input is maximized, and the transport enhances the assignment. Therefore, after a cycle, there is little room for a change of the optimal mask when the optimization is performed again.

C. Hyperparameter Settings

C.1. CIFAR-100

We train models via stochastic gradient descent (SGD) with initial learning of 0.1 decayed by factor 0.1 at epochs 200 and 300 for WRN28-10 and epochs 400 and 800 for PreActResNet18. We set the momentum as 0.9 and add a weight decay of 0.0001. Mixing weight λ is randomly sampled from $Beta(1, 1)$ for all experiments except Manifold mixup, which uses $Beta(2, 2)$ in the original paper. Puzzle Mix has hyperparameters of β for the label smoothness term, γ for the data smoothness term, η for the prior term, and ξ for the transport cost. In the CIFAR-100 experiment, we use $(\beta, \gamma, \eta, \xi) = (1.2, 0.5, 0.2, 0.8)$. For adversarial training, we use 10/255 epsilon-ball with the step size τ of 12/255 according to the step size protocol of Wong et al. (2020).

C.2. Tiny-ImageNet

We follow the training protocol of Verma et al. (2019) except for the learning schedule. Verma et al. (2019) train Tiny-ImageNet for 2000 epochs with an initial learning rate of 0.1, but we train models for 1200 epochs with an initial learning rate of 0.2. As in the CIFAR-100 experiment, we use SGD and decay learning rate by factor 0.1 at epochs 600 and 900. We set momentum as 0.9 and weight decay as 0.0001. In case of mixing weight λ, for Input mixup and Manifold mixup, we follow the setting $\alpha = 0.2$ as described in Manifold mixup (Verma et al., 2019). For CutMix, we choose $\alpha = 0.2$, which showed the best performance among [0.2, 0.5, 1.0], and for Puzzle Mix, we use $\alpha = 1.0$. In the Tiny-ImageNet experiment, we use $(\beta, \gamma, \eta, \xi) = (1.2, 0.5, 0.2, 0.8)$, which is the same with the CIFAR-100 experiment. However, we apply regularization using clean input with $\lambda_{clean} = 1$ for all experiments regarding Puzzle Mix, and use the same initial learning rate of 0.2 for *Puzzle Mix (half)*.

C.3. ImageNet

For ImageNet, we modify the training protocol in Wong et al. (2020) and train models for 100 epochs. The learning rate starts from 0.5, linearly increases to 1.0 for 8 epochs, and linearly decreases to 0.125 until 15$^{\text{th}}$ epoch. Then, the learning rate jumps to 0.2 and linearly decreases to 0.02 until 40$^{\text{th}}$ epoch, 0.002 until 65$^{\text{th}}$ epoch, 0.0002 until 90$^{\text{th}}$ epoch, and 0.00002 until 100$^{\text{th}}$ epoch. In addition, we resize images to 160×160 for the first 15 epochs, and use images pre-resized to 352×352 for the next 85 epochs following the prescription in Wong et al. (2020). Mixing distribution parameter α is 0.2, 0.2, 1.0, 1.0 each for Input mixup, Manifold mixup, CutMix, Puzzle Mix, which follows the settings

of the original papers. In the case of Manifold mixup, there is no experiments on ImageNet, and thus, we tune α in [0.2, 1.0] and report the best result. In the case of ImageNet, we use hyperparameter $(\beta, \gamma, \eta, \xi) = (1.5, 0.5, 0.2, 0.8)$ and apply clean input regularization of $\lambda_{clean} = 1$ for the first 40 epochs.

C.4. Hyperparameter Sensitivity

We analyze the sensitivity of the hyperparameters with WRN28-10 on CIFAR100 trained for 400 epochs. In detail, we sweep hyperparameters one by one while others being fixed, and calculate the mean and standard deviation of Top-1 accuracy. Note that, the hyperparameter setting $(\beta, \gamma, \eta, \xi)$ of the main experiment is (1.2, 0.5, 0.2, 0.8) which achieves 15.95% Top-1 test error.

Table 1 shows the mean Top-1 error rates and standard deviations of various hyperparameter settings. From the table, we can find that there exists a well of hyperparameters of which performance is superior to that of baselines (manifold mixup: 17.40%).

Parameter	Range	Mean Top-1 Error(%) (SD)
β	[0.8, 1.6]	16.19% (0.22)
γ	[0.0, 1.0]	16.43% (0.20)
η	[0.1, 0.35]	16.37% (0.18)
ξ	[0.4, 1.0]	16.25% (0.27)

Table 1. Mean Top-1 error rates and standard deviations (SD) for various hyperparameter settings on CIFAR 100 with WRN28-10. For β, γ, and ξ, we sweep the range with 0.1 step size, and for η, we sweep the range with 0.05 step size.

D. Effect of Adversarial Training

D.1. Trade-off between Generalization and Adversarial Robustness

Since adversarial training increases the adversarial robustness at the expense of clean accuracy (Madry et al., 2017), we introduced adversarial probability p, a probability of whether to add adversarial perturbation or not, to control the intensity of adversarial training. Table 2 shows the inverse relationship between clean error and FGSM error.

D.2. Robustness against PGD Attack

We test the adversarial robustness of various mixup methods against the PGD attack (Madry et al., 2017). In this experiment, we train PreActResNet18 on CIFAR-100 with each mixup method and test with PGD attack of 4/255 l_∞ epsilon-ball with 2/255 step size. For comparison, we test Puzzle Mix with stochastic adversarial training of $p = 0.1$, which outperforms other baselines at Top-1 Accuracy given a clean test dataset. Figure 3 demonstrates that Puzzle Mix is more robust against the PGD attack than the existing mixup methods.

Adversarial Probability	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
0.00	37.58	19.40	92.70
0.05	37.60	19.10	89.12
0.10	37.16	19.25	86.09
0.15	38.14	19.70	83.91
0.20	38.65	20.01	82.25
0.25	39.46	20.40	80.37
0.30	40.52	21.47	79.76

Table 2. Top-1 / Top-5 / FGSM error rates on Tiny-ImageNet dataset for PreActResNet18 trained with various adversarial probability p.

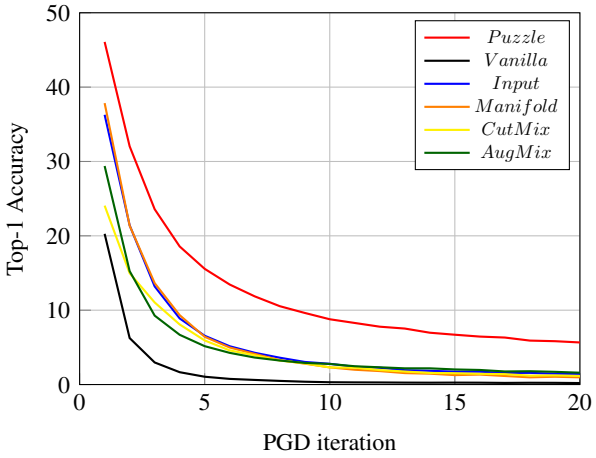

Figure 3. Adversarial robustness of various mixup methods against the PGD attack.

E. Puzzle Mix Qualitative Results

E.1. Effect of Prior and Smoothness Term

In this section, we provide Puzzle Mix results while adjusting the hyperparameters associated with the optimal mask. In Figure 4, we visualize how the Puzzle Mix images change by increasing the mixing weight λ, and in Figure 5, we visualize how the results change as we increase the coefficients of the smoothness terms, β and γ.

E.2. More Samples

In this section, we provide Puzzle Mix results with various resolutions of the optimal mask and transport. Figure 6 visualizes the Puzzle Mix results along with the given inputs.

References

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. *NeurIPS*, 2017.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Courville, A., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. *ICML*, 2019.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

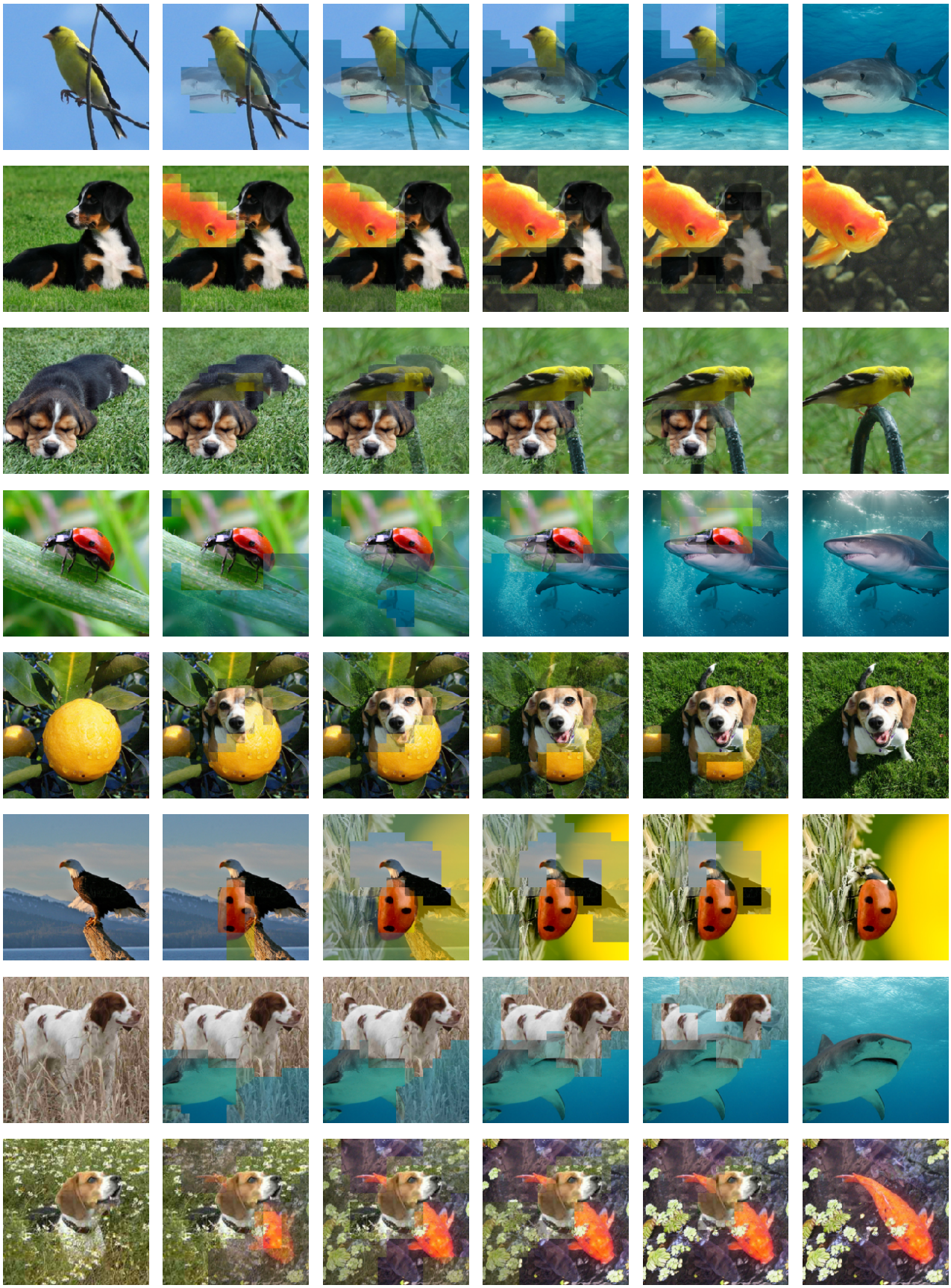

Figure 4. Puzzle Mix images with increasing mixing weight λ.

Figure 5. Puzzle Mix images with increasing smoothness coefficient β and γ.

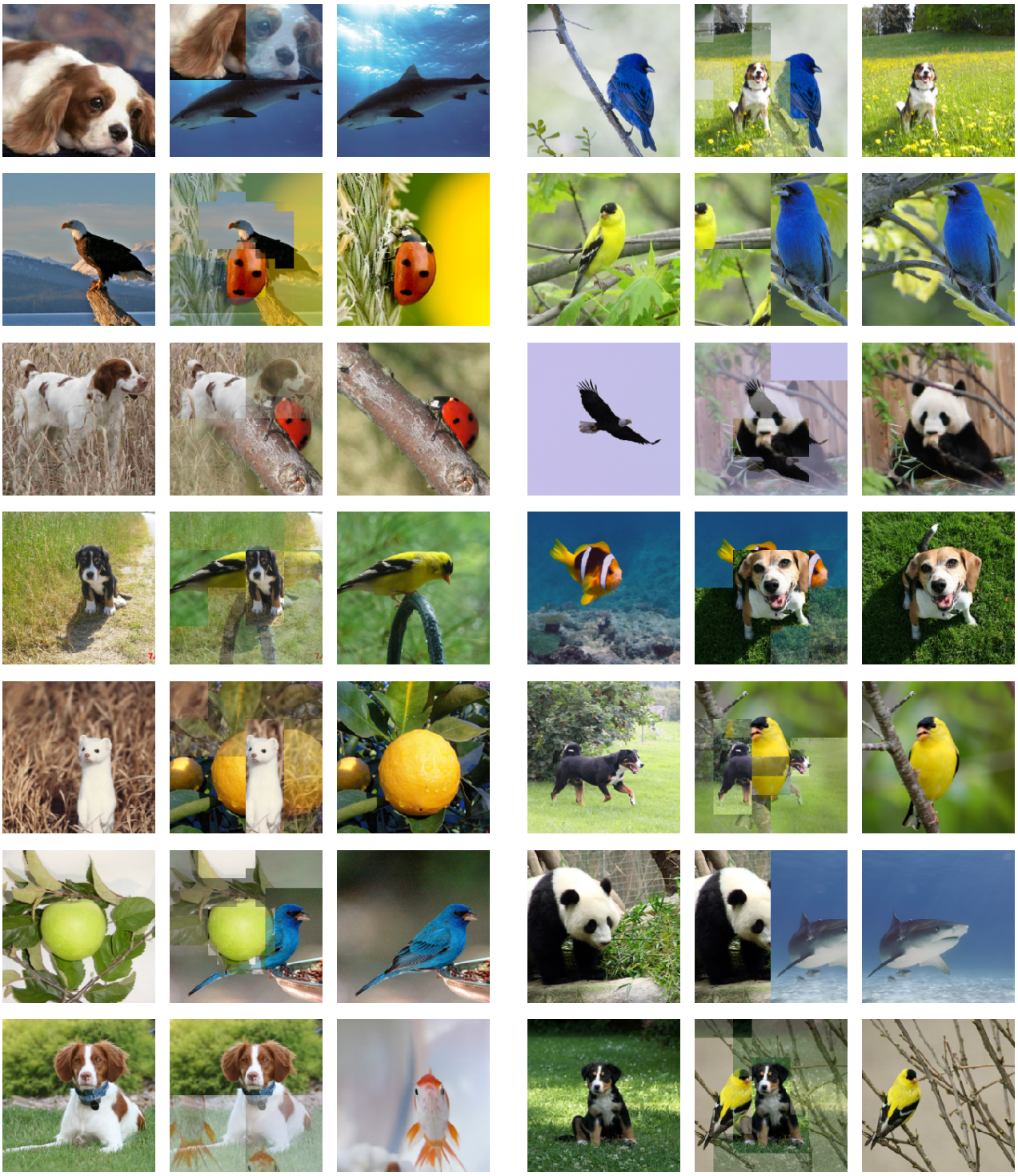

Figure 6. Various Puzzle Mix image samples. Each row consists of input image 1 (left), Puzzle Mix image (middle), and input image 2 (right).