Supplementary: DART: Diversify-Aggregate-Repeat Training Improves Generalization of Neural Networks

Samyak Jain [*] [◇] [‡] Sravanti Addepalli [*]
Pawan Kumar Sahu [∓] [§] [‡] Priyam Dey [∓] R.Venkatesh Babu
Vision and AI Lab, Indian Institute of Science, Bangalore

1. Theoretical Results

In this section, we present details on the theoretical results discussed in Section-5 of the main paper. As noted by Shen *et al*. [16], the weights learned by a patch-wise Convolutional Neural Network are a linear combination of the two types of features (described in Section-5 of the main paper) present in the dataset. Let the threshold K_{cut} denote the number of robust features learned by the model. We have,

$$w = \sum_{k \leq K_{cut}} v_k + \sum_{k > K_{cut}} y^{(k)} \epsilon^{(k)} \tag{1}$$

On averaging of the weights of m models we get:

$$w = \frac{1}{m} \sum_{j=1}^{m} \left[\sum_{k=1}^{K_{cut_j}} v_{k_j} + \sum_{k > K_{cut_j}} y_j^{(k)} \epsilon_j^{(k)} \right] \tag{2}$$

We now analyze the convergence of this weight averaged neural network shown in Eq.2. Let L represent the logistic loss of the model, F denote the function learned by the neural network, and w_c denote its weights across C channels indexed using c. Further, let $y^{(i)}$ represent the ground truth label of sample $x_i \ \forall \ i \in [1, n]$, where n denotes the number of samples in the train set. The weights $w_1, w_2, .., w_C$ are initialized as $w_c \sim \mathcal{N}\left(0, \sigma_0^2 I_d\right) \ \forall \ c \in C$. We assume that the weights learned by the model at any time stamp t are a linear combination of the linear functions f, g and h corresponding to feature patches, noisy patches and model initialization respectively, as shown below:

$$w_c^t = f(v_1, v_2, \dots v_K) + g(\epsilon^{(1)}, \epsilon^{(2)}, \dots \epsilon^{(n)}) + h(\epsilon') \tag{3}$$

where ϵ' is the random noise sampled for the initialization of the model. Since the term $h(\epsilon')$ does not play a role in the convergence of the model, we ignore this term for the purpose of analysis. For simplicity, we assume that f and g represent summations over their respective arguments. Thus, the weights at any time t can be represented as

$$w_c^t = \sum_{l=1}^{K_{cut}^t} \alpha_l^t v_l + \sum_{l > K_{cut}^t} y^{(l)} \epsilon^{(l)} \tag{4}$$

where K_{cut}^t and α_l^t are a functions of time t. At convergence, $\alpha_i = 1 \ \forall i \in [1, K_{cut}]$ and $\alpha_i = 0$ otherwise.

We now analyze the learning dynamics while training the model. Owing to the gradient descent based updates of model weights over time, the derivative of overall loss L w.r.t. the weights of a given channel w_c can be written as,

$$\begin{aligned} \frac{d}{dt} w_c &= -\frac{d}{dw_c} L \\ &= -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} L^{'}(y^{(i)}, F(w, x^{(i)})) \nabla_{w_c} F(w, x^{(i)}) \end{aligned} \tag{5}$$

Since L is a logistic loss, we have $-L^{'}(o(1)) = 0.5 + o(1)$, where $o(1)$ represents terms independent of the variable w. As discussed in Section-5 of the main paper, the function learned by the neural network is given by $F(w, x) = \sum_{c=1}^{C} \sum_{p=1}^{2} \phi(w_c, x_p)$, where ϕ is the activation function defined as follows [16]:

- for $|z| \leq 1$; $\phi(z) = sign(z) \frac{1}{q} |z|^q$
- for $z \geq 1$; $\phi(z) = z - \frac{q-1}{q}$
- for $z \leq -1$; $\phi(z) = z + \frac{q-1}{q}$

Based on this, Eq.5 can be written as

$$\frac{d}{dt} w_c \approx \frac{1 + o(1)}{2n} \sum_{i=1}^{n} \sum_{p=1}^{2} \phi^{'}(|w_c x_p^{(i)}|) y^{(i)} x_p^{(i)} \tag{6}$$

[*]Equal Contribution. [∓] Equal contribution second authors. Correspondence to Samyak Jain <samyakjain.cse18@itbhu.ac.in>, Sravanti Addepalli <sravantia@iisc.ac.in>. [◇] Indian Institute of Technology, Varanasi [§] Indian Institute of Technology, Dhanbad. [‡] Work done during internship at Vision and AI Lab, Indian Institute of Science, Bangalore.

Considering the two types of patches present in the image (feature and noisy patch), we have:

$$\frac{d}{dt}w_c \approx \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c v_{d^{(i)}}|)v_{d^{(i)}} + \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c \epsilon^{(i)}|)y^{(i)}\epsilon^{(i)} \quad (7)$$

where $v_{d^{(i)}}$ represents the feature patch in the image $x^{(i)}$, $\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c v_{d^{(i)}}|)v_{d^{(i)}}$ represents the gradients on feature patches, and $\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c \epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}$ represents the gradients on noisy patches of the image.

To improve the clarity of the proofs, we restate and proof lemma-1 of [16] in the following two lemmas presented below:

Lemma 1 *Let $X \sim N(0, \sigma_x^2 I_m)$ and $Y \sim N(0, \sigma_y^2 I_m)$ be m dimensional gaussian random variables, then $X^T Y = O(\sqrt{m}\sigma_x\sigma_y)$*

Proof. Given any two random variables $x \sim N(\mu_1, \sigma_1^2)$ and $y \sim N(\mu_2, \sigma_2^2)$

$$Var(xy) = E[x^2y^2] - E[(xy)]^2 = Var(x)Var(y) + Var(x)E(y)^2 + Var(y)E(x)^2 = \sigma_1^2\sigma_2^2 + \sigma_1^2\mu_2^2 + \sigma_2^2\mu_1^2 \quad (8)$$

For $\mu_1 = \mu_2 = 0$, we get

$$Var(xy) = Var(x)Var(y) \quad (9)$$

Let $X \sim N(0, \sigma_x^2 I_m)$ and $Y \sim N\left(0, \sigma_y^2 I_m\right)$ be m dimensional gaussian random variables, *i.e.*, $X = [x_0, x_1, x_2, ..., x_{m-1}]$ and $Y = [y_0, y_1, y_2, ..., y_{m-1}]$, where $x_i \sim N\left(0, \sigma_x^2\right)$ and $y_i \sim N\left(0, \sigma_y^2\right)$ $\forall i \in \{0, 1, 2, ..., m-1\}$. Calculating $Var(X^T Y)$,

$$Var(X^T Y) = E\left[\left(X^T Y\right)^2\right] = E\left[\left(\sum_{i=0}^{m-1} x_i y_i\right)^2\right] \quad (10)$$

Since each x_i and y_i are sampled *i.i.d* from a Gaussian with a fixed mean and variance, therefore the product $x_i y_i$ is also an *i.i.d* random variable with a distribution of the difference of two chi-squared distributions. The sum of such k chi-squared random variables with mean μ and variance σ^2 results in a chi-squared distribution with mean $k\mu$ and variance $k\sigma^2$. Given this, let $z = X^T Y$. Therefore, by Eq.9, z has a zero mean and a variance of $m\sigma_x^2\sigma_y^2$. Thus, we have

$$Var(z) = E(z^2) = m\sigma_x^2\sigma_y^2 \quad (11)$$

Using Chebyshev's inequality, we have

$$P(|z| \geq k\sqrt{m}\sigma_x\sigma_y) \leq \frac{1}{k^2} \quad (12)$$

where k is some constant. Therefore, we have

$$z = O\left(\sqrt{m}\sigma_x\sigma_y\right) \quad (13)$$

Further, by central limit theorem, we have the distribution of $z = X^T Y = \sum_{i=0}^{m-1} x_i y_i$ to be approximately Gaussian. Therefore, even for a small value of k, we have a high confidence interval for bounding $|z|$. □

Lemma 2 *Let V be a standard basis vector and $Y \sim N(0, \sigma_y^2 I_m)$ be N dimensional gaussian random variable, then $V^T Y = O(\sigma_y)$*

Proof. Let $Y \sim N(0, \sigma_y^2 I_m)$ be m-dimensional gaussian random variable, *i.e.*, $Y = [y_0, y_1, y_2, ..., y_{m-1}]$ where each $y_i \sim N(0, \sigma_y^2)$ $\forall i \in \{0, 1, 2, ..., m-1\}$. Let $V = [v_0, v_1, v_2, ..., v_{m-1}]$ and $z = V^T Y$. Since V is a standard basis vector, we have

$$Var(z) = E\left[\left(V^T Y\right)^2\right] = E\left[\left(\sum_{i=0}^{m-1} v_i y_i\right)^2\right] = E\left[\left(y_k\right)^2\right] = Var(y_k) = \sigma_y^2 \quad (14)$$

where k is some index for which $v_k = 1$ and $v_j = 0$ $\forall j \neq k$. Using Chebyshev's inequality, we have

$$P(|z| \geq k\sigma_y) \leq \frac{1}{k^2} \quad (15)$$

where k is some constant. Therefore we have

$$z = O(\sigma_y) \quad (16)$$

□

Based on the above lemmas, considering the weights $w_c \sim (0, \sigma_0^2 I_d)$, we have

$$|w_c v_k| = O(\sigma_0) \quad (17)$$

$$|w_c \epsilon^{(i)}| = O(\sigma\sigma_0) \quad (18)$$

$$|\epsilon^{(j)}\epsilon^{(i)}| = O\left(\frac{\sigma^2}{\sqrt{d}}\right) \quad (19)$$

$$|\epsilon^{(i)} v_k| = O\left(\frac{\sigma}{\sqrt{d}}\right) \quad (20)$$

1.1. Convergence time for feature patches

Data Augmentations: As defined by Shen *et al.* [16], an augmentation T_k can be defined as follows:

$$\forall\ k' \in [1,K],\quad \mathcal{T}_k(v_{k'}) = v_{((k'+k-1)\ mod\ K)+1} \tag{21}$$

Assuming that K unique augmentation strategies are used (where K denotes the number of robust patches in the dataset), augmented data is defined as follows:

$$D_{train}^{(aug)} = D_{train} \cup \mathcal{T}_1(D_{train}) .. \cup \mathcal{T}_{K-1}(D_{train}) \tag{22}$$

where D_{train} is the training dataset. This ensures that each feature patch v_i appears n times in the dataset, thus making the distribution of all the feature patches uniform. In the proposed method, we consider that m models are being independently trained after which their weights are averaged as shown below:

$$w = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{K_{cut_j}} v_{k_j} + \frac{1}{m}\sum_{j=1}^{m}\sum_{k>K_{cut_j}} y_j^{(k)}\epsilon_j^{(k)} \tag{23}$$

Each branch is trained on the dataset $D_{train}^{(k)}$ defined as:

$$D_{train}^{(k)} = \mathcal{T}_k(D_{train}),\ \ k \in [1,2,...,m] \tag{24}$$

Proposition 1 *The convergence time for learning any feature patch v_i $\forall i \in [1,K]$ in at least one channel $c \in C$ of the weight averaged model f_θ using the augmentations defined in Eq.24, is given by $O\left(\frac{K}{\sigma_0^{q-2}}\right)$, if $\frac{\sigma^q}{\sqrt{d}} \ll \frac{1}{K}$, $m = K$.*

Proof. We first compute the convergence time without weight-averaging, as shown by Shen *et al.* [16]. The dot product between $\frac{dw_c}{dt}$ (from Eq.7) and any given feature v_k is given by:

$$\frac{d}{dt}w_c \cdot v_k \approx \frac{1+o(1)}{2}\rho_k \phi^{'}(|w_c v_k|) + \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c\epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}v_k \tag{25}$$

where, ρ_k represents the fraction of v_k in the dataset. At initialization, we have $w_c \sim (0, \sigma_0^2 I_d)$. Therefore, using conditions at initialization in Eq.17, 18 and 20 along with the definition of the activation function defined for the case $|w_c v_{d^{(i)}}| < 1$ and $|w_c\epsilon^{(i)}| < 1$, we arrive at the following convergence time for the feature and the noisy patch, respectively:

$$\frac{1+o(1)}{2n}\phi^{'}(|w_c v_k|) = O\left(\rho_k \sigma_0^{q-1}\right) \tag{26}$$

$$\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c\epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}v_k = O\left(\frac{\sigma_0^{q-1}\sigma^q}{\sqrt{d}}\right) \tag{27}$$

A closer look at the above two equations reveal that if $\frac{\sigma^q}{\sqrt{d}} \ll \frac{1}{K}$, the noisy patch term in Eq.25 (the second term) can be ignored in comparison to the feature patch term (the first term). This gives:

$$\frac{d}{dt}w_c \cdot v_k \approx \frac{1+o(1)}{2}\rho_k \phi^{'}(|w_c v_k|) \tag{28}$$

Let us denote the term $w_c \cdot v_k$ at any time step t using a generic function $g \equiv g(w_c, v_k, t)$. Using the definition of the activation function ϕ, and assuming that $|w_c v_k| < 1$, we get

$$\frac{dg}{dt} = \frac{1+o(1)}{2}\rho_k g^{q-1} \tag{29}$$

On integrating, we get the following:

$$\frac{(1+o(1))\rho_k}{2}(2-q)t + g(t=0)^{2-q} = g(t=t)^{2-q} \tag{30}$$

$$t = O\left(\frac{1}{\rho_k \sigma_0^{q-2}}\right) \tag{31}$$

We now compute the convergence of the case where m models are averaged. We denote the averaged weights of a given channel c by w_c^{avg}. By substituting for w_c from Eq.4, we get

$$-\frac{1}{m}\sum_{j=1}^{m}\left(\frac{dL}{dw_c}\right)_j v_k = \frac{dw_c^{avg}}{dt}v_k = \frac{1}{m}\sum_{j=1}^{m}\frac{d}{dt}\left(\sum_{l=1}^{K}\alpha_{lj}^t v_l + \sum_{l>K_{cut}^t} y_j^{(l)}\epsilon_j^{(l)}\right)v_k \tag{32}$$

Using $|\epsilon^{(i)}v_k| = O\left(\frac{\sigma}{\sqrt{d}}\right)$ from Eq.20 gives us $\sum_{l>K_{cut}^t} y^{(l)}\epsilon^{(l)}v_k = O\left(\frac{\sigma}{\sqrt{d}}\right)$, whereas $\sum_{l=1}^{K}\alpha_{lj}^t v_l = O(1)$. Since d represents the number of parameters, we can say $d \gg \sigma$. Further, since $\epsilon^{(l)}$ are *i.i.d* random variables, therefore, the value of the noise component $\sum_{l>K_{cut}^t} y^{(l)}\epsilon^{(l)}v_k$ is expected to further decrease upon averaging over m models. Thus, ignoring it w.r.t. to the feature term $\sum_{l=1}^{K}\alpha_{lj}^t v_l$, we get

$$\frac{dw_c^{avg}}{dt}v_k \approx \frac{1}{m}\frac{d}{dt}\left(\sum_{j=1}^{m}\alpha_{kj}^t\right) \tag{33}$$

A similar analysis for a single model that is not weight-averaged gives

$$\frac{dw_c}{dt}v_k \approx \frac{d\alpha_k^t}{dt} = \frac{dw_c^{avg}}{dt}v_k \frac{d\left(m\alpha_k^t\right)}{d\left(\sum_{j=1}^{m}\alpha_{kj}^t\right)} \tag{34}$$

As discussed in Section-5 of the main paper, we set $m = K$. Further, since the most frequent patches are learned faster, we assume that the relative rate of change in α_{kj} will depend on the relative frequency of individual patch features. Therefore, $\frac{d\alpha_k^t/dt}{d\left(\sum_{j=1}^{m}\alpha_{kj}^t\right)/dt} = \frac{d\alpha_k^t/dt}{d\left(\sum_{j=1}^{K}\alpha_{kj}^t\right)/dt} = \rho_k$. Thus we get,

$$\frac{dw_c^{avg}}{dt}v_k = \frac{1}{\rho_k K}\frac{dw_c}{dt}v_k \tag{35}$$

In Eq.35, we have the rate of change of $w_c^{avg} = \frac{1}{\rho_k K}$ times the rate of change of w_c. Therefore the time for convergence for w_c^{avg} will be $\rho_k K$ times the time for convergence for w_c, which gives

$$t = O\left(\frac{K}{\sigma_0^{q-2}}\right) \tag{36}$$

□

Corollary 1.1 *The convergence time for learning any feature patch v_i $\forall i \in [1, K]$ in at least one channel $c \in C$ of the weight averaged model f_θ using the augmentations defined in Eq.24, is given by $O\left(\frac{m\rho_k^{'}}{\rho_k\sigma_0^{q-2}}\right)$, if $\frac{\sigma^q}{\sqrt{d}} \ll \frac{1}{K}$.*

Here ρ_k is the ratio between the frequency of the feature patch k in the dataset and the sum of the frequencies of all feature patches in the dataset. $\rho_k^{'}$ is the ratio between the frequency of the feature patch k in the dataset and the sum of the frequencies of some m feature patches $[v_{(k) \ mod \ K+1}, v_{(k+1) \ mod \ K+1}, ..., v_{(m+k-1) \ mod \ K+1}]$

Proof. Since the most frequent patches are learned faster, we assume that the relative rate of change in α_{kj} will depend on the relative frequency of individual patch features. Therefore,

$$\frac{d\alpha_k^t/dt}{d\left(\sum_{j=1}^{m}\alpha_{kj}^t\right)/dt} = \frac{(\alpha_k^t)/dt}{\left(\sum_{j=1}^{m}\alpha_{kj}^t\right)/dt} = \rho_k^{'} \tag{37}$$

Thus substituting in Eq.34, we get

$$\frac{dw_c^{avg}}{dt}v_k = \frac{1}{\rho_k^{'}m}\frac{dw_c}{dt}v_k \tag{38}$$

In Eq.38, we have the rate of change of $w_c^{avg} = \frac{1}{\rho_k^{'}m}$ times the rate of change of w_c. Therefore, the time for convergence for w_c^{avg} will be $\rho_k^{'}m$ times the time for convergence for w_c, which gives

$$t = O\left(\frac{m\rho_k^{'}}{\rho_k\sigma_0^{q-2}}\right) \tag{39}$$

□

The convergence time from corollary-1.1 (denoted as t) can be written as

$$t = O\left(\frac{m\sum_{j=1}^{K}\alpha_{kj}}{\sum_{j=1}^{m}\alpha_{kj}\sigma_0^{q-2}}\right) \tag{40}$$

The convergence time from Eq.31 (denoted as $t^{'}$) can be written as

$$t^{'} = O\left(\frac{\sum_{j=1}^{K}\alpha_{kj}}{\alpha_k\sigma_0^{q-2}}\right) \tag{41}$$

For hard to learn feature patches (feature patches with low α_k), upon comparing Eq.40 and Eq.41, we observe that the convergence time will be higher in Eq.41. Since a summation over some m feature patches is appearing in Eq.40, therefore its convergence time has a lower impact on the frequency of an individual feature patch. This helps in enhanced learning of hard features, thereby improving generalization.

1.2. Convergence time of noisy patches

We consider the dot product between any noisy patch ϵ^k and Eq.7:

$$\frac{d}{dt}w_c\epsilon^{(k)} = \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c v_{d^{(i)}}|)v_{d^{(i)}}\epsilon^{(k)} + \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c\epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}\epsilon^{(k)} \tag{42}$$

On simplifying we get,

$$\frac{d}{dt}w_c\epsilon^{(k)} = \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c v_{d^{(i)}}|)v_{d^{(i)}}\epsilon^{(k)} + \frac{1+o(1)}{2n}\phi^{'}(|w_c\epsilon^{(k)}|)y^{(k)}||\epsilon^{(k)}||^2 + \frac{1+o(1)}{2n}\sum_{i=1;i\neq k}^{n}\phi^{'}(|w_c\epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}\epsilon^{(k)} \tag{43}$$

In Eq.43 we can ignore $\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_c v_{d^{(i)}}|)v_{d^{(i)}}\epsilon^{(k)} + \frac{1+o(1)}{2n}\sum_{i=1;i\neq k}^{n}\phi^{'}(|w_c\epsilon^{(i)}|)y^{(i)}\epsilon^{(i)}\epsilon^{(k)}$ as compared to $\frac{1+o(1)}{2n}\phi^{'}(|w_c\epsilon^{(k)}|)y^{(k)}||\epsilon^{(k)}||^2$, if their values are of different orders at initialization. Since, at initialization, $w_c \sim (0, \sigma_0^2 I_d)$, using conditions in Eq.17-20 and the definition of

the activation function defined for the case of $|w_c v_{d^{(i)}}| < 1$ and $|w_c \epsilon^{(i)}| < 1 \forall i \in [1, n]$, we get

$$\frac{1+o(1)}{2n} \sum_{i=1}^{n} \phi^{'}(|w_c v_{d^{(i)}}|) v_{d^{(i)}} \epsilon^{(k)} = O\left(\frac{\sigma_0^{q-1}\sigma}{\sqrt{d}}\right) \quad (44)$$

$$\frac{1+o(1)}{2n} \sum_{i=1;i\neq k}^{n} \phi^{'}(|w_c \epsilon^{(i)}|) y^{(i)} \epsilon^{(i)} \epsilon^{(k)} = O\left(\frac{\sigma_0^{q-1}\,\sigma^{q+1}}{\sqrt{d}}\right) \quad (45)$$

Therefore,

$$\frac{1+o(1)}{2n} \sum_{i=1}^{n} \phi^{'}(|w_c v_{d^{(i)}}|) v_{d^{(i)}} \epsilon^{(k)} + \frac{1+o(1)}{2n} \sum_{i=1;i\neq k}^{n} \phi^{'}(|w_c \epsilon^{(i)}|) y^{(i)} \epsilon^{(i)} \epsilon^{(k)} = O\left(\frac{\sigma_0^{q-1}\,\sigma^{q+1}}{\sqrt{d}}\right) + O\left(\frac{\sigma_0^{q-1}\sigma}{\sqrt{d}}\right) \quad (46)$$

$$\frac{1+o(1)}{2n} \phi^{'}(|w_c \epsilon^{(k)}|) y^{(k)} ||\epsilon^{(k)}||^2 = \frac{1+o(1)}{2n} \sigma^2 \phi^{'}(|w_c \epsilon^{(k)}|) y^{(k)} = O\left(\frac{\sigma^{q+1}\sigma_0^{q-1}}{n}\right) \quad (47)$$

Comparing Eq.47 and Eq.46, we get if $d \gg n^2$, we can ignore the term $\frac{1+o(1)}{2n} \sum_{i=1}^{n} \phi^{'}(|w_c v_{d^{(i)}}|) v_{d^{(i)}} \epsilon^{(k)} + \frac{1+o(1)}{2n} \sum_{i=1;i\neq k}^{n} \phi^{'}(|w_c \epsilon^{(i)}|) y^{(i)} \epsilon^{(i)} \epsilon^{(k)}$ as compared to $\frac{1+o(1)}{2n} \phi^{'}(|w_c \epsilon^{(k)}|) y^{(k)} ||\epsilon^{(k)}||^2$ Thus, we get the following:

Using the activation defined earlier, and considering the value of $w_c \epsilon^{(k)}$ at time stamp t given by $g(w_c, \epsilon^{(k)}, t)$, where $|g(w_c, \epsilon^{(k)}, t)| < 1$, we get

$$\frac{d(g(w_c, \epsilon^{(k)}, t))}{dt} = \frac{1+o(1)}{2n} \sigma^2 g(w_c, \epsilon^{(k)}, t)^{q-1} \quad (48)$$

Similar to the analysis presented in Eq.30, on integrating the above equation, we get

$$\frac{1+o(1)}{2n}(2-q)t\sigma^2 + g(w_c, \epsilon^{(k)}, t=0)^{2-q} = g(w_c, \epsilon^{(k)}, t=t)^{2-q} \quad (49)$$

Using Eq.18, at $t = 0$,

$$g(w_c, \epsilon^{(k)}, t=0)^{2-q} = \sigma_0^{2-q}\sigma^{2-q} \quad (50)$$

where σ_0 is the standard deviation of the zero-mean Gaussian distribution that is used for initializing the weights of the model, and $\frac{\sigma}{\sqrt{d}}$ is the standard deviation of the noise present in noisy patches. Thus, we get

$$\frac{1+o(1)}{2n}(2-q)t\sigma^2 + \sigma_0^{2-q}\sigma^{2-q} = g(w_c, \epsilon^{(k)}, t=t)^{2-q} \quad (51)$$

At the time of convergence, the term $g(w_c, \epsilon^{(k)}, t = t)^{2-q}$ will become $o(1)$. Therefore, $\frac{1+o(1)}{2n}(2-q)t\sigma^2 + \sigma_0^{2-q}\sigma^{2-q}$ should be constant. Equating the L.H.S. of the above equation to 0, the convergence time to learn $\epsilon^{(k)}$ by at least one channel $c \in C$ is given by:

$$t = O\left(\frac{n}{\sigma_0^{q-2}\sigma^q}\right) \quad (52)$$

Proposition 2 *If the noise patches learned by each f_θ^k are $i.i.d.$ Gaussian random variables $\sim \mathcal{N}\left(0, \frac{\sigma^2}{d} I_d\right)$ then with high probability, convergence time of learning a noisy patch $\epsilon^{(j)}$ in at least one channel $c \in [1, C]$ of the weight averaged model f_θ is given by $O\left(\frac{nm}{\sigma_0^{q-2}\sigma^q}\right)$, if $d \gg n^2$.*

Proof. By averaging the weights of m models in Eq.42, we get

$$-\frac{1}{m}\sum_{j=1}^{m}\left(\frac{dL}{dw_c}\right)_j \epsilon^{(k)} = \frac{dw_{avg}}{dt}\epsilon^{(k)} = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_{c_j} v_{d^{(i)}}|) v_{d^{(i)}}\epsilon^{(k)} + \frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_{c_j}\epsilon^{(i)}|) y^{(i)}\epsilon^{(i)}\epsilon^{(k)}\right] \quad (53)$$

$$= \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1+o(1)}{2n}\sum_{i=1}^{n}\phi^{'}(|w_{c_j} v_{d^{(i)}}|) v_{d^{(i)}}\epsilon^{(k)} + \frac{1+o(1)}{2n}\phi^{'}(|w_{c_j}\epsilon^{(k)}|) y^{(k)}||\epsilon^{(k)}||_2^2 + \frac{1+o(1)}{2n}\sum_{i=1;i\neq k}^{n}\phi^{'}(|w_{c_j}\epsilon^{(i)}|) y^{(i)}\epsilon^{(i)}\epsilon^{(k)}\right] \quad (54)$$

$$= \sum_{j=1}^{m}\frac{1}{m}\left[\frac{1+o(1)}{2n}\sigma^2\phi^{'}(|w_{c_j}\epsilon^{(k)}|) y^{(k)} + \tau\right] \quad (55)$$

where τ consists of the remaining terms that are negligible since the noise learned by each model is $i.i.d.$ and $d \gg n^2$.

Using the weights learned by different models as represented in Eq.4, we get,

$$\frac{dw_{avg}}{dt}\epsilon^{(k)} \approx \frac{1+o(1)}{2n}\sigma^2 y^{(k)}\frac{1}{m}\sum_{j=1}^{m}\phi^{'}\big(|(\sum_{l=1}^{K}\alpha_{lj}^{t}v_l + \sum_{l>K_{cut}^{t}} y_j^{(l)}\epsilon_j^{(l)})\epsilon^{(k)}|\big) \quad (56)$$

Since the noise $\epsilon^{(i)}$ learned by different models is considered as $i.i.d.$, we get

$$\frac{dw_{avg}}{dt}\epsilon^{(k)} = \frac{1+o(1)}{2nm}\Bigg(\sigma^2 y^{(k)}\phi^{'}\big(|\sum_{l=1}^{K}\alpha_{lk}^{t}v_l\epsilon^{(k)} + \sum_{l>K_{cut}^{t}} y_k^{(l)}\sigma^2|\big) + \sum_{i=1;i\neq k}^{m}\phi^{'}\big(|\sum_{l=1}^{K}\alpha_{li}^{t}v_l\epsilon^{(k)} + \sum_{l>K_{cut}^{t}} y_i^{(l)}\epsilon_i^{(l)}\epsilon^{(k)}|\big)\Bigg) \quad (57)$$

Note that from Eq.19, we have $\sum_{l>K_{cut}^{t}} y^{(l)}\epsilon_i^{(l)}\epsilon^{(k)} = O\left(\frac{\sigma^2}{\sqrt{d}}\right)$, and from Eq.20, we get $\sum_{i=1;i\neq k}^{n}\phi^{'}\left(\left|\sum_{l=1}^{K}\alpha_{li}^{t}v_l\epsilon^{(k)}\right|\right) = O\left(\left(\frac{\sigma}{\sqrt{d}}\right)^{q-1}\right)$. Whereas $y_k^{(l)}\sigma^2 = O(1)$. Since it is assumed that $d \gg n^2$, therefore, we can ignore the terms $\sum_{l=1}^{K}\alpha_{li}^{t}v_l\epsilon^{(k)}$ and $\sum_{l>K_{cut}^{t}} y^{(l)}\epsilon_i^{(l)}\epsilon^{(k)}$ in comparison to $\sum_{l>K_{cut}^{t}} y_k^{(l)}\sigma^2$. Thus, we get

$$\frac{dw_{avg}}{dt}\epsilon^{(k)} = \frac{1+o(1)}{2nm}\sigma^2 y^{(k)}\phi^{'}\big(|\sum_{l=1}^{K}\alpha_{lk}^{t}v_l\epsilon^{(k)} + \sum_{l>K_{cut}^{t}} y_k^{(l)}\sigma^2|\big) \quad (58)$$

Similarly, we derive the learning dynamics of a single model w_{c_k} below:

$$\frac{dw_{c_k}}{dt}\epsilon^{(k)} = \frac{1+o(1)}{2n}\sigma^2 y^{(k)}\phi^{'}\big(|\sum_{l=1}^{K}\alpha_{lk}^{t}v_l\epsilon^{(k)} + \sum_{l>K_{cut}^{t}} y_k^{(l)}\sigma^2|\big) \quad (59)$$

From Eq.58 and Eq.59, we get the following relation

$$\frac{1}{m}\frac{dw_{c_k}}{dt}\epsilon^{(k)} = \frac{dw_{avg}}{dt}\epsilon^{(k)} \quad (60)$$

In Eq.60, we have the rate of change of w_{avg} equals $\frac{1}{m}$ times the rate of change of w_{c_k}. Therefore, the time for convergence for w_{avg} will be m times the time for convergence for w_{c_k}, which gives the convergence time for learning the noisy patch, $\epsilon^{(k)}$ by at least one channel $c \in C$ of the model as

$$t = O\left(\frac{nm}{\sigma_0^{q-2}\sigma^q}\right) \quad (61)$$

□

Proposition 3 *If the noise learned by each f_θ^k are $i.i.d.$ Gaussian random variables $\sim \mathcal{N}\left(0, \frac{\sigma^2}{d}I_d\right)$, and model weight averaging is performed at epoch T, the convergence time of learning a noisy patch $\epsilon^{(j)}$ in at least one channel $c \in [1, C]$ of the weight averaged model f_θ is given by $T + O\left(\frac{nm^{(q-2)}d^{(q-2)/2}}{\sigma^{(2q-2)}}\right)$, if $d \gg n^2$.*

Proof. We assume that the model is close to convergence at epoch T. Hence, its weights can be assumed to be similar to Eq.2. Further, we assume that the weights are composed of noisy and feature patches as shown in Eq.2. Since the noisy patches are assumed to be $i.i.d.$, the standard deviation of the weights corresponding to noisy features is given by $\frac{\sigma}{m\sqrt{d}}$. Thus, using the above lemmas, we get

$$g(w_c, \epsilon^{(k)}, t = T)^{2-q} = \sigma^{4-2q}m^{q-2}d^{\frac{q-2}{2}} \quad (62)$$

On integrating Eq.48 from time T and substituting the above, we get

$$\frac{1+o(1)}{2n}t(2-q)\sigma^2 + \sigma^{4-2q}m^{q-2}d^{\frac{q-2}{2}} = g(w_c, \epsilon^{(k)}, t = t)^{2-q} \quad (63)$$

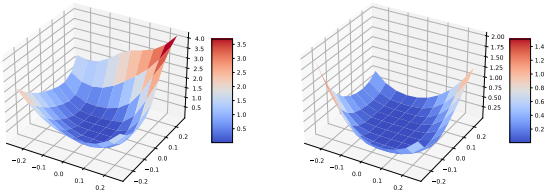
Thus, the convergence time of learning at least one channel $c \in C$ by on using this initialization is given by

$$t = O\left(\frac{nm^{(q-2)}d^{(q-2)/2}}{\sigma^{(2q-2)}}\right) \quad (64)$$

Further, the total convergence time is given by

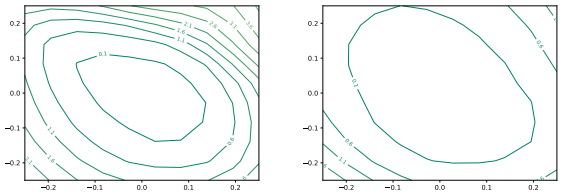
$$T + O\left(\frac{nm^{(q-2)}d^{(q-2)/2}}{\sigma^{(2q-2)}}\right) \quad (65)$$

Since we have considered the weights to be composed of two parts and the model is assumed to be converged with respect to feature patches, therefore, using such an initialization will not impact their learning dynamics. □



(a) ERM+EMA (PCH) (b) DART (Ours, PCH)

Figure 1. **Loss landscape visualization**



(a) ERM+EMA (PCH) (b) DART (Ours, PCH)

Figure 2. **Loss Contour visualization**

1.3. Impact of intermediate interpolations

We assume that T in Proposition-3 is negligible w.r.t. $O\left(\frac{nm^{(q-2)}d^{(q-2)/2}}{\sigma^{(2q-2)}}\right)$. We further analyze the ratio of the convergence time from Proposition-3 (denoted as t) and Proposition-2 (denoted as t'),

$$\frac{t}{t'} = O\left(\frac{m^{q-3}d^{(q-2)/2}\sigma_0^{q-2}}{\sigma^{q-2}}\right) \tag{66}$$

A lower bound on the above equation will occur when $d = n^2$ and $q = 3$. Using this, we get

$$\frac{t}{t'} = O\left(\frac{n\sigma_0}{\sigma}\right) \tag{67}$$

Thus, the lower bound is of the order n which is greater than 1. Therefore, the convergence time of learning a noisy patch $\epsilon^{(j)}$ in at least one channel $c \in [1, C]$ on performing an intermediate interpolation (Prop.3) is greater than the case where weight-averaging of only final models is performed (Prop.2), by upto $O(n)$.

2. Loss surface plots

We compare the loss surface of the proposed method with ERM training on CIFAR-100 dataset using WRN-28-10 architecture. To exclusively understand the impact of the proposed Diversify-Aggregate-Repeat steps, we present results using the simple augmentations - Pad and Crop followed by Horizontal Flip (PCH) for both ERM and DART. We use exponential moving averaging (EMA) of weights in both the ERM baseline and DART for a fair comparison.

Table 1. **Loss Landscape Sharpness Analysis:** Comparison of the proposed method DART (Pad-Crop) and ERM (Pad-Crop) trained using WRN-28-10 on CIFAR-100. The metrics presented here have been adapted from Stutz *et al.* [17]. For all metrics, a lower value corresponds to a flatter loss landscape.

Method	**Worst Case Flatness** ↓	**Average Flatness** ↓	**Average Train Loss** ↓
ERM (Pad+Crop)	4.173	1.090	0.0028
DART (Pad+Crop) (Ours)	**2.037**	**0.294**	**0.0022**

As shown in Fig.1, the loss surface of the proposed method DART is flatter when compared to the ERM baseline. The same is also evident from the level sets of the contour plot in Fig.2. In Table-1, we also use the scale-invariant metrics proposed by Stutz *et al.* [17] to quantitatively verify that the flatness of loss surface is indeed better using the proposed approach DART. Worst Case Flatness represents the Cross-Entropy loss on perturbing the weights in an ℓ_2 norm ball of radius 0.25. Average Flatness represents the Cross-Entropy loss on adding random Gaussian noise with standard deviation 0.25, and further clamping it so that the added noise remains within the ℓ_2 norm ball of radius 0.25. Average Train Loss represents the loss on train set images as shown in Table-1. We achieve lower values when compared to the ERM baseline across all metrics, demonstrating that the proposed method DART has a flatter loss landscape compared to ERM.

3. Additional Results: ID generalization

3.1. Model coefficients

While in the proposed method DART, we give equal weight to all M branches, we note that fine-tuning the weights of individual models in a greedy manner [21] can give a further boost in accuracy. As shown in Fig.3, the best accuracy obtained is 86.33% at $\lambda_1 = 0.17$, $\lambda_2 = 0.46$, when compared to 86.24% with $\lambda_1 = \lambda_2 = 0.33$. These results are lower than those reported in Table-2 of the main paper and Table-3 in the supplementary since the runs in Fig.3 do not use EMA, while our main method does.

3.2. Training plots

We show the training plots for In-domain generalization training of CIFAR-100 on WRN-28-10 in Fig.4. We firstly note that not only does our method yield gains on the final interpolation step (as seen in Table-3 and Table-2 of the main paper), but the step of intermediate interpolation ensures that the individual models are also better than the ERM baselines trained using the respective augmentations. Specifically, while the initial interpolations help in bringing the models closer to each other in the loss landscape, the later ones actually result in performance gains, since the low

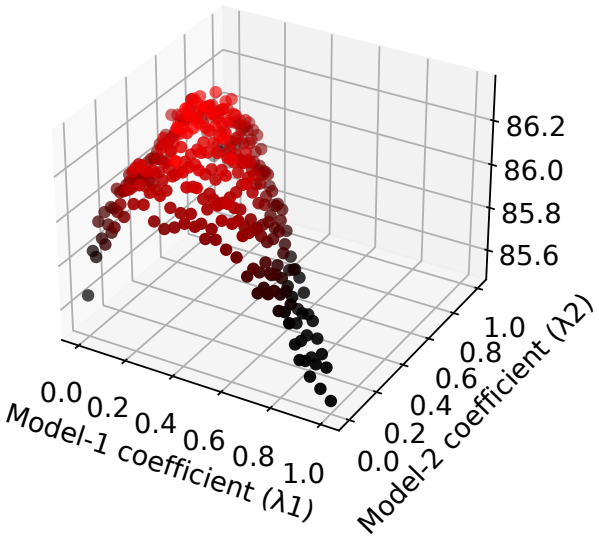


Figure 3. **Accuracy (%) on interpolating the final converged models trained using DART (ours)** using WRN-28-10 model and CIFAR-100 dataset, by taking their convex combination. Maximum accuracy of 86.33 is obtained on interpolating, using three experts with accuracies 85.65, 85.75 and 85.51. For the best setting, $\lambda_1 = 0.17$ and $\lambda_2 = 0.46$.

Table 2. **Integrating DART with SAM** gives around 0.2% improvement in performance (%) when compared to SAM with mixed augmentations. The results are shown on CIFAR-100 dataset using WRN-28-10 model.

ERM+EMA	ERM+SWA	DART	SAM+EMA	DART+SAM+EMA
85.57 $\pm$ 0.13	85.44 $\pm$ 0.09	86.46 $\pm$ 0.12	87.05 $\pm$ 0.15	**87.26** $\pm$ 0.02

learning rate ensures that the flatter loss surface obtained using intermediate weight averaging is retained.

3.3. Integrating DART with SAM

Table-2 shows that the proposed approach DART integrates effectively with SAM to obtain further performance gains. However, the gains are relatively lower on integrating with SAM ($\sim 0.2\%$) when compared to the gains over Mixed ERM training ($\sim 0.9\%$). We hypothesize that this is because SAM already encourages smoothness of loss surface, which is also achieved using DART.

3.4. Evaluation across different model capacities

We present results of DART on ResNet-18 and WideResNet-28-10 models in Table-3. The gains obtained on WideResNet-28-10 are larger (0.2 and 0.89) when compared to ResNet-18 (0.06 and 0.64) demonstrating the scalability of our method.

Table 3. **Different model architectures:** Performance (%) of DART when compared to Mixed-Training (MT) across different architectures. Standard deviation is reported across 5 reruns.

Model	Method	CIFAR-10	CIFAR-100
ResNet18	ERM+EMA (Mixed - MT)	97.08 $\pm$ 0.05	82.25 $\pm$ 0.29
	DART (Ours)	**97.14** $\pm$ 0.08	**82.89** $\pm$ 0.07
WRN-28-10	ERM+EMA (Mixed - MT)	97.76 $\pm$ 0.17	85.57 $\pm$ 0.13
	DART (Ours)	**97.96** $\pm$ 0.06	**86.46** $\pm$ 0.12

4. Details on Domain Generalization

4.1. Training Details

Since the domain shift across individual domains is larger in the Domain Generalization setting when compared to the In-Domain generalization setting, we found that training individual branches on a mix of all domains was better than training each branch on a single domain. Moreover, training on a mix of all domains also improves the individual branch accuracy, thereby boosting the accuracy of the final interpolated model. We train 4 branches (6 for DomainNet), where one branch is trained with an equal proportion of all domains, while the other three branches are allowed to be experts on individual domains by using a higher fraction (40% for DomainNet and 50% for other datasets) of the selected domain for the respective branch.

In the Domain Generalization setting, the step of explicitly training on mixed augmentations / domains (L4-L5 in Algorithm-1 in the main paper) is replaced by the initialization of the model using ImageNet pretrained weights, which ensures that all models are in the loss basin. Moreover, this also helps in reducing the overall compute.

For the results presented in the Tables 4.3.1 to 4.3.6 and Table-4 of the main paper, the training configuration (training iterations, interpolation frequency) was set to (15k, 1k) for DomainNet and (10k, 1k) for all other datasets, whereas for the results presented in Table-5 in the main paper, the configuration was set to (5k, 600) for DANN [6] and CDANN [12], and (8k, 1k) for the rest, primarily to reduce compute. The difference in adversarial training approaches (DANN and CDANN) was primarily because their training is not stable for longer training iterations. SWAD-specific hyperparameters were set as suggested by the authors [4] without additional tuning. While we compare with comparable compute for the In-Domain generalization setting (Table-3 and Table-2 in the main paper), for the Domain Generalization setting we report the baselines from DomainBed [7] and the respective papers as is the common practice. Although the proposed approach uses higher compute than the baselines, we show in Fig.5(a) that even with higher compute, the baselines cannot achieve any better performance.

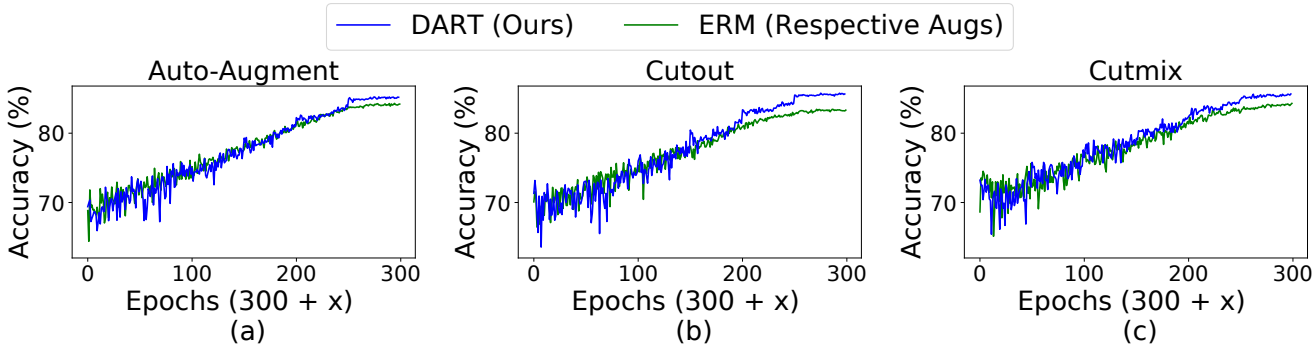


Figure 4. **Comparison of the test accuracy (%) of ERM training using different augmentations with the respective augmentation expert of DART** on CIFAR-100, WRN-28-10. The analysis is done from 300 epochs onwards. Most of the gains of the proposed method occur at the end of training, when learning rate is low and the experts are present within a common basin.

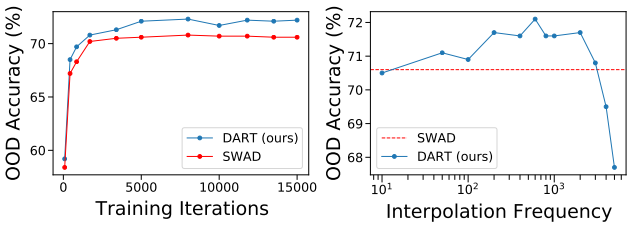


Figure 5. **Performance of DART across (a) varying training iterations and (b) varying interpolation frequency:** (a) compares the proposed method DART's performance with the SWAD baseline when trained for higher number of iterations. Interpolation frequency was maintained such that the number of interpolations remained same (8) in every case. (b) demonstrates the effect of intermediate interpolation frequency on DART. The training iterations were kept constant (5k).

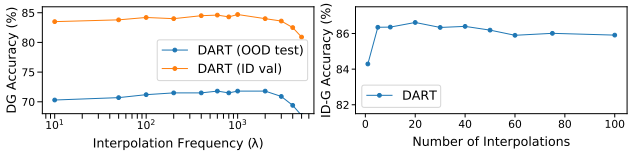


Figure 6. **Study on hyperparameter sensitivity for DART** (a) shows variation in the accuracy of DART for different interpolation frequencies on the OfficeHome dataset using ResNet-50 model with ImageNet initialization. A strong correlation between the in and out-of-domain accuracy of DART in the DG setting is observed. (b) shows the variation of In Domain Accuracy for the CIFAR-100 dataset and WRN-28-10 model vs. the number of interpolations done during the training. It is seen that the in-domain accuracy is stable across a wide range of interpolations.

4.2. Ablation experiments

We present ablation experiments on the Office-Home dataset in Fig.5. Following this, we present average accuracy across all domain splits as is common practice in Domain Generalization [7].

Variation across training compute: Fig.5 (a) demonstrates that the performance of SWAD plateaus early compared to DART, when trained for a higher number of training iterations. We note that DART achieves a significant improvement in the final accuracy over the baseline. This indicates that although the proposed method requires higher compute, DART trades it off for improved performance.

Variation in interpolation frequency: Fig.5 (b) describes the impact of varying the interpolation frequency in the proposed method DART. The number of training iterations is set to 5k for this experiment. We note that the accuracy is stable across a wide range of interpolation frequencies (x-axis is in log scale). This shows that the proposed method is not very sensitive to the frequency of interpolation, and does not require fine-tuning for every dataset. We therefore use the same interpolation frequency of 1k for all the datasets and training splits of DomainBed. We note that the proposed method performs better than baseline in all cases except when the frequency is kept too low (≈10) or too high (close to total training iterations). The sharp deterioration in performance in the case of no intermediate interpolation (interpolation frequency = total training steps) illustrates the necessity of intermediate interpolation in the proposed method.

4.3. Detailed Results

In this section, we present complete Domain Generalization results (Out-of-domain accuracies in %) on VLCS (Table-4.3.2), PACS (Table-4.3.3), OfficeHome (Table-4.3.4), TerraIncognita (Table-4.3.5) and DomainNet (Table-4.3.6) benchmarks. We also present the average accuracy across all domain splits and datasets in Table-4.3.1. We note that the proposed method DART when combined with SWAD [4] outperforms all existing methods across all datasets.

4.3.1 Averages

Algorithm	VLCS	PACS	OfficeHome	TerraIncognita	DomainNet	Avg
ERM [19]	77.5 ± 0.4	85.5 ± 0.2	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	63.3
IRM [1]	78.5 ± 0.5	83.5 ± 0.8	64.3 ± 2.2	47.6 ± 0.8	33.9 ± 2.8	61.6
GroupDRO [14]	76.7 ± 0.6	84.4 ± 0.8	66.0 ± 0.7	43.2 ± 1.1	33.3 ± 0.2	60.7
Mixup [20]	77.4 ± 0.6	84.6 ± 0.6	68.1 ± 0.3	47.9 ± 0.8	39.2 ± 0.1	63.4
MLDG [10]	77.2 ± 0.4	84.9 ± 1.0	66.8 ± 0.6	47.7 ± 0.9	41.2 ± 0.1	63.6
CORAL [18]	78.8 ± 0.6	86.2 ± 0.3	68.7 ± 0.3	47.6 ± 1.0	41.5 ± 0.1	64.5
MMD [11]	77.5 ± 0.9	84.6 ± 0.5	66.3 ± 0.1	42.2 ± 1.6	23.4 ± 9.5	58.8
DANN [6]	78.6 ± 0.4	83.6 ± 0.4	65.9 ± 0.6	46.7 ± 0.5	38.3 ± 0.1	62.6
CDANN [12]	77.5 ± 0.1	82.6 ± 0.9	65.8 ± 1.3	45.8 ± 1.6	38.3 ± 0.3	62.0
MTL [3]	77.2 ± 0.4	84.6 ± 0.5	66.4 ± 0.5	45.6 ± 1.2	40.6 ± 0.1	62.9
SagNet [13]	77.8 ± 0.5	86.3 ± 0.2	68.1 ± 0.1	48.6 ± 1.0	40.3 ± 0.1	64.2
ARM [22]	77.6 ± 0.3	85.1 ± 0.4	64.8 ± 0.3	45.5 ± 0.3	35.5 ± 0.2	61.7
VREx [9]	78.3 ± 0.2	84.9 ± 0.6	66.4 ± 0.6	46.4 ± 0.6	33.6 ± 2.9	61.9
RSC [8]	77.1 ± 0.5	85.2 ± 0.9	65.5 ± 0.9	46.6 ± 1.0	38.9 ± 0.5	62.7
SWAD [4]	79.1 ± 0.1	88.1 ± 0.1	70.6 ± 0.2	50.0 ± 0.3	46.5 ± 0.1	66.9
DART w/o SWAD	78.5 ± 0.7	87.3 ± 0.5	70.1 ± 0.2	48.7 ± 0.8	45.8	66.1
DART w/ SWAD	**80.3** ± 0.2	**88.9** ± 0.1	**71.9** ± 0.1	**51.3** ± 0.2	**47.2**	**67.9**

4.3.2 VLCS

Algorithm	C	L	S	V	Avg
ERM	97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5
IRM	98.6 ± 0.1	64.9 ± 0.9	73.4 ± 0.6	77.3 ± 0.9	78.5
GroupDRO	97.3 ± 0.3	63.4 ± 0.9	69.5 ± 0.8	76.7 ± 0.7	76.7
Mixup	98.3 ± 0.6	64.8 ± 1.0	72.1 ± 0.5	74.3 ± 0.8	77.4
MLDG	97.4 ± 0.2	65.2 ± 0.7	71.0 ± 1.4	75.3 ± 1.0	77.2
CORAL	98.3 ± 0.1	66.1 ± 1.2	73.4 ± 0.3	77.5 ± 1.2	78.8
MMD	97.7 ± 0.1	64.0 ± 1.1	72.8 ± 0.2	75.3 ± 3.3	77.5
DANN	**99.0** ± 0.3	65.1 ± 1.4	73.1 ± 0.3	77.2 ± 0.6	78.6
CDANN	97.1 ± 0.3	65.1 ± 1.2	70.7 ± 0.8	77.1 ± 1.5	77.5
MTL	97.8 ± 0.4	64.3 ± 0.3	71.5 ± 0.7	75.3 ± 1.7	77.2
SagNet	97.9 ± 0.4	64.5 ± 0.5	71.4 ± 1.3	77.5 ± 0.5	77.8
ARM	98.7 ± 0.2	63.6 ± 0.7	71.3 ± 1.2	76.7 ± 0.6	77.6
VREx	98.4 ± 0.3	64.4 ± 1.4	74.1 ± 0.4	76.2 ± 1.3	78.3
RSC	97.9 ± 0.1	62.5 ± 0.7	72.3 ± 1.2	75.6 ± 0.8	77.1
SWAD	98.8 ± 0.1	63.3 ± 0.3	75.3 ± 0.5	79.2 ± 0.6	79.1
DART w/o SWAD	97.9 ± 1.0	64.2 ± 0.7	73.9 ± 1.1	78.1 ± 1.6	78.5
DART w/ SWAD	98.7 ± 0.0	**66.4** ± 0.3	**75.8** ± 0.6	**80.4** ± 0.3	**80.3**

4.3.3 PACS

Algorithm	A	C	P	S	Avg
ERM	84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5
IRM	84.8 ± 1.3	76.4 ± 1.1	96.7 ± 0.6	76.1 ± 1.0	83.5
GroupDRO	83.5 ± 0.9	79.1 ± 0.6	96.7 ± 0.3	78.3 ± 2.0	84.4
Mixup	86.1 ± 0.5	78.9 ± 0.8	97.6 ± 0.1	75.8 ± 1.8	84.6
MLDG	85.5 ± 1.4	80.1 ± 1.7	97.4 ± 0.3	76.6 ± 1.1	84.9
CORAL	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2
MMD	86.1 ± 1.4	79.4 ± 0.9	96.6 ± 0.2	76.5 ± 0.5	84.6
DANN	86.4 ± 0.8	77.4 ± 0.8	97.3 ± 0.4	73.5 ± 2.3	83.6
CDANN	84.6 ± 1.8	75.5 ± 0.9	96.8 ± 0.3	73.5 ± 0.6	82.6
MTL	87.5 ± 0.8	77.1 ± 0.5	96.4 ± 0.8	77.3 ± 1.8	84.6
SagNet	87.4 ± 1.0	80.7 ± 0.6	97.1 ± 0.1	80.0 ± 0.4	86.3
ARM	86.8 ± 0.6	76.8 ± 0.5	97.4 ± 0.3	79.3 ± 1.2	85.1
VREx	86.0 ± 1.6	79.1 ± 0.6	96.9 ± 0.5	77.7 ± 1.7	84.9
RSC	85.4 ± 0.8	79.7 ± 1.8	97.6 ± 0.3	78.2 ± 1.2	85.2
DMG [5]	82.6	78.1	94.3	78.3	83.4
MetaReg [2]	87.2	79.2	97.6	70.3	83.6
DSON [15]	87.0	80.6	96.0	82.9	86.6
SWAD	89.3 ± 0.2	83.4 ± 0.6	97.3 ± 0.3	78.2 ± 0.5	88.1
DART w/o SWAD	87.1 ± 1.5	83.5 ± 0.9	96.9 ± 0.3	81.8 ± 0.9	87.3
DART w/ SWAD	**90.1** ± 0.1	**84.5** ± 0.2	**97.7** ± 0.2	**83.4** ± 0.1	**88.9**

4.3.4 OfficeHome

Algorithm	A	C	P	R	Avg
ERM	61.3 ± 0.7	52.4 ± 0.3	75.8 ± 0.1	76.6 ± 0.3	66.5
IRM	58.9 ± 2.3	52.2 ± 1.6	72.1 ± 2.9	74.0 ± 2.5	64.3
GroupDRO	60.4 ± 0.7	52.7 ± 1.0	75.0 ± 0.7	76.0 ± 0.7	66.0
Mixup	62.4 ± 0.8	54.8 ± 0.6	76.9 ± 0.3	78.3 ± 0.2	68.1
MLDG	61.5 ± 0.9	53.2 ± 0.6	75.0 ± 1.2	77.5 ± 0.4	66.8
CORAL	65.3 ± 0.4	54.4 ± 0.5	76.5 ± 0.1	78.4 ± 0.5	68.7
MMD	60.4 ± 0.2	53.3 ± 0.3	74.3 ± 0.1	77.4 ± 0.6	66.3
DANN	59.9 ± 1.3	53.0 ± 0.3	73.6 ± 0.7	76.9 ± 0.5	65.9
CDANN	61.5 ± 1.4	50.4 ± 2.4	74.4 ± 0.9	76.6 ± 0.8	65.8
MTL	61.5 ± 0.7	52.4 ± 0.6	74.9 ± 0.4	76.8 ± 0.4	66.4
SagNet	63.4 ± 0.2	54.8 ± 0.4	75.8 ± 0.4	78.3 ± 0.3	68.1
ARM	58.9 ± 0.8	51.0 ± 0.5	74.1 ± 0.1	75.2 ± 0.3	64.8
VREx	60.7 ± 0.9	53.0 ± 0.9	75.3 ± 0.1	76.6 ± 0.5	66.4
RSC	60.7 ± 1.4	51.4 ± 0.3	74.8 ± 1.1	75.1 ± 1.3	65.5
SWAD	66.1 ± 0.4	57.7 ± 0.4	78.4 ± 0.1	80.2 ± 0.2	70.6
DART w/o SWAD	64.3 ± 0.2	57.9 ± 0.9	78.3 ± 0.6	79.9 ± 0.1	70.1
DART w/ SWAD	**67.1** ± 0.2	**59.2** ± 0.1	**79.7** ± 0.1	**81.5** ± 0.1	**71.9**

4.3.5 TerraIncognita

Algorithm	L100	L38	L43	L46	Avg
ERM	49.8 ± 4.4	42.1 ± 1.4	56.9 ± 1.8	35.7 ± 3.9	46.1
IRM	54.6 ± 1.3	39.8 ± 1.9	56.2 ± 1.8	39.6 ± 0.8	47.6
GroupDRO	41.2 ± 0.7	38.6 ± 2.1	56.7 ± 0.9	36.4 ± 2.1	43.2
Mixup	59.6 ± 2.0	42.2 ± 1.4	55.9 ± 0.8	33.9 ± 1.4	47.9
MLDG	54.2 ± 3.0	44.3 ± 1.1	55.6 ± 0.3	36.9 ± 2.2	47.7
CORAL	51.6 ± 2.4	42.2 ± 1.0	57.0 ± 1.0	39.8 ± 2.9	47.6
MMD	41.9 ± 3.0	34.8 ± 1.0	57.0 ± 1.9	35.2 ± 1.8	42.2
DANN	51.1 ± 3.5	40.6 ± 0.6	57.4 ± 0.5	37.7 ± 1.8	46.7
CDANN	47.0 ± 1.9	41.3 ± 4.8	54.9 ± 1.7	39.8 ± 2.3	45.8
MTL	49.3 ± 1.2	39.6 ± 6.3	55.6 ± 1.1	37.8 ± 0.8	45.6
SagNet	53.0 ± 2.9	43.0 ± 2.5	57.9 ± 0.6	40.4 ± 1.3	48.6
ARM	49.3 ± 0.7	38.3 ± 2.4	55.8 ± 0.8	38.7 ± 1.3	45.5
VREx	48.2 ± 4.3	41.7 ± 1.3	56.8 ± 0.8	38.7 ± 3.1	46.4
RSC	50.2 ± 2.2	39.2 ± 1.4	56.3 ± 1.4	**40.8** ± 0.6	46.6
SWAD	55.4 ± 0.0	44.9 ± 1.1	59.7 ± 0.4	39.9 ± 0.2	50.0
DART w/o SWAD	54.6 ± 1.1	44.9 ± 1.6	58.7 ± 0.5	36.6 ± 1.9	48.7
DART w/ SWAD	**56.3** ± 0.4	**47.1** ± 0.3	**61.2** ± 0.3	40.5 ± 0.1	**51.3**

4.3.6 DomainNet

Algorithm	clip	info	paint	quick	real	sketch	Avg
ERM	58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9
IRM	48.5 ± 2.8	15.0 ± 1.5	38.3 ± 4.3	10.9 ± 0.5	48.2 ± 5.2	42.3 ± 3.1	33.9
GroupDRO	47.2 ± 0.5	17.5 ± 0.4	33.8 ± 0.5	9.3 ± 0.3	51.6 ± 0.4	40.1 ± 0.6	33.3
Mixup	55.7 ± 0.3	18.5 ± 0.5	44.3 ± 0.5	12.5 ± 0.4	55.8 ± 0.3	48.2 ± 0.5	39.2
MLDG	59.1 ± 0.2	19.1 ± 0.3	45.8 ± 0.7	13.4 ± 0.3	59.6 ± 0.2	50.2 ± 0.4	41.2
CORAL	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5
MMD	32.1 ± 13.3	11.0 ± 4.6	26.8 ± 11.3	8.7 ± 2.1	32.7 ± 13.8	28.9 ± 11.9	23.4
DANN	53.1 ± 0.2	18.3 ± 0.1	44.2 ± 0.7	11.8 ± 0.1	55.5 ± 0.4	46.8 ± 0.6	38.3
CDANN	54.6 ± 0.4	17.3 ± 0.1	43.7 ± 0.9	12.1 ± 0.7	56.2 ± 0.4	45.9 ± 0.5	38.3
MTL	57.9 ± 0.5	18.5 ± 0.4	46.0 ± 0.1	12.5 ± 0.1	59.5 ± 0.3	49.2 ± 0.1	40.6
SagNet	57.7 ± 0.3	19.0 ± 0.2	45.3 ± 0.3	12.7 ± 0.5	58.1 ± 0.5	48.8 ± 0.2	40.3
ARM	49.7 ± 0.3	16.3 ± 0.5	40.9 ± 1.1	9.4 ± 0.1	53.4 ± 0.4	43.5 ± 0.4	35.5
VREx	47.3 ± 3.5	16.0 ± 1.5	35.8 ± 4.6	10.9 ± 0.3	49.6 ± 4.9	42.0 ± 3.0	33.6
RSC	55.0 ± 1.2	18.3 ± 0.5	44.4 ± 0.6	12.2 ± 0.2	55.7 ± 0.7	47.8 ± 0.9	38.9
MetaReg	59.8	**25.6**	50.2	11.5	64.6	50.1	43.6
DMG	65.2	22.2	50.0	15.7	59.6	49.0	43.6
SWAD	66.0 ± 0.1	22.4 ± 0.3	53.5 ± 0.1	**16.1** ± 0.2	65.8 ± 0.4	55.5 ± 0.3	46.5
DART w/o SWAD	65.9	21.9	52.6	15.1	64.9	54.3	45.8
DART w/ SWAD	**66.5**	22.8	**54.2**	**16.1**	**67.3**	**56.3**	**47.2**

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 10

[2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems (NeurIPS)*, 31, 2018. 10

[3] Gilles Blanchard, Aniket Anand Deshmukh, Ürun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *The Journal of Machine Learning Research (JMLR)*, 22(1):46–100, 2021. 10

[4] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:22405–22418, 2021. 8, 9, 10

[5] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *European Conference on Computer Vision (ECCV)*, pages 301–318. Springer, 2020. 10

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 17(59):1–35, 2016. 8, 10

[7] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations (ICLR)*, 2021. 8, 9

[8] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision (ECCV)*, pages 124–140. Springer, 2020. 10

[9] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning (ICML)*, pages 5815–5826. PMLR, 2021. 10

[10] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, 2018. 10

[11] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5400–5409, 2018. 10

[12] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, 2018. 8, 10

[13] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8690–8699, 2021. 10

[14] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations (ICLR)*, 2020. 10

[15] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *European Conference on Computer Vision (ECCV)*. Springer, 2020. 10

[16] Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. Data augmentation as feature manipulation: a story of desert cows and grass cows. In *International Conference on Machine Learning (ICML)*, 2022. 1, 2, 3

[17] David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7807–7817, 2021. 7

[18] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision (ECCV)*, pages 443–450. Springer, 2016. 10

[19] Vladimir Vapnik. Statistical learning theory wiley. *New York*, 1998. 10

[20] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020. 10

[21] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, pages 23965–23998. PMLR, 2022. 7

[22] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:23664–23678, 2021. 10