A. Pretraining Details

In this section, we delve into the implementation details of our QD4V pretraining approach to provide a comprehensive understanding of our methodology. Firstly, we present a PyTorch style code for our branched ResNet ensemble in Listing 1. We also present a code snippet to calculate the QD loss (Eq. 7) using this branched ensemble in Listing 2 for a single mini-batch.

Model initialisation: We initialise the models with ImageNet pretrained weights obtained from Pytorch's torchvision package for QD4V supervised pre-training. Note that Listing 1 snippet shows the model initialisation only for supervised pre-training. For MoCo pre-training, we use a MoCov2 pretrained checkpoint trained for 800 epochs from the official implementation of [15]. For experiments with ImageNet100, we use the ImageNet-100 pretrained checkpoint available in the official implementation of [38]. Both query and momentum encoder is initialised with the query and momentum encoder respectively from official implementation.

Formulating batches for QD4V: As shown in Eq 3, the formulation of $|\mathcal{T}|$-dimensional phenotype requires features for $|\mathcal{T}|$ transformed inputs. This leads to additional forward passes through each member of the ensemble leading to increased computational overhead. To tackle this issue, we design a PyTorch dataloader that concatenates augmented and unaugmented mini-batches of size m such that all features required to evaluate the phenotype can be evaluated with a single forward pass. For example, given an unaugmented batch $[x_1, x_2, ..., x_m]$ and transformed inputs $[[\mathcal{T}_1(x_1), ..., \mathcal{T}_1(x_m)], [\mathcal{T}_2(x_1), ..., \mathcal{T}_2(x_m)], \cdots]$, the dataloader combines them into one single batch, such that we can perform a single pass to obtain $|\mathcal{T}| + 1$ outputs (logits) and features. This is also shown in Listing 2. Outputs corresponding to unaugmented images (or weakly augmented in case of MoCo) are used for quality loss.

Hyperparameters: For all downstream experiments in Table 1, Table 2, and Table 3, we consider an ensemble with $N = 5$ members. For supervised pre-training, we train the model for 20 epochs with SGD optimizer with a batch-size of 128. We adopt a learning rate warm-up for 5 epochs with a warmup decay of 0.01, followed by a cosine decay schedule with learning rate of 0.5. For supervised pre-training, we also apply label smoothening and perform exponential moving average (EMA) on the model weights. For contrastive learning experiments, we use the SGD optimizer with a learning a rate of 0.03 with a batch size of 128. For both pre-training paradigms, we set λ_d is set to 0.2 and λ_{kl} is set to 0.1.

Training competitor - Diverse Baseline In Table 1, Table 2, and Table 3, we evaluate our supervised pre-training approach by comparing it to a diverse ensemble strategy (referred to as the 'div' ensemble) which was inspired by concepts from [23, 50]. The purpose of these ensemble strategies is to increase the robustness of predictions by promoting diversity among the predictions of the different members in the ensemble. In this section, we provide details on how we implemented this baseline and trained it.

We denote a training sample by (x, y), where y can be obtained from ground truth or self-supervision. Similar to our experiments, we introduce a population of N feature extractors, represented as an ensemble, $\mathcal{F} = \{f_{\theta_i}\}_{i=1}^N$, which aim to learn a diverse set of probability distributions. We denote the predictions of a model by $\hat{y}_i = \sigma(g_{\phi_i}(f_{\theta_i}(x)))$, where g_{ϕ_i} is the projection head or classifier layer corresponding to i^{th} member of the ensemble and σ denotes the softmax function. To incorporate diversity in predictions, [23, 50] employ symmetrised KL divergence between predictions of the ensemble. We borrow this concept and formulate a loss term corresponding to diversity shown in Eq.

$$L_{\text{diversity,KL}} = \sum_{i \neq j}^{N} \frac{1}{2} \Big(\text{KL}(y_i \| y_j; \tau) + \text{KL}(y_j \| y_i; \tau) \Big) \quad (9)$$

Here, τ denotes the temperature used to scale the logits within softmax. Thus, the total loss function becomes

$$\mathcal{L}_{qd} = \mathcal{L}_{\text{quality}} + \lambda_d \mathcal{L}_{\text{diversity,KL}} \quad (10)$$

The resulting loss function in Eq. 10 encourages the networks to make the right prediction according to the ground-truth label, but then they are also encouraged to make different second-best, third-best, and so on, predictions.

Similar to our the formulation of ensembles for QD4V pre-training, we share the first three layers (`layer1`, `layer2`, `layer3`, after which different members of the ensemble branch out into their own sequence of layers having a separate `layer4` and projection head. We also initialise the entire ensemble with ImageNet pre-trained models. We train the model for 20 epochs with SGD optimizer, along with learning rate warm-up for 5 epochs, followed by a cosine decay schedule. For these experiments, λ_d is set to 0.5 and τ is set to 6.0.

B. Downstream tasks

Section 5 in our paper shows extensive evaluation of QD4V pre-training on a diverse set of classification, regession and dense estimation tasks. Complete details on datasets and tasks are provided in Table 5.

We provide a code snippet to find best model parameters $\{\tilde{U}_i\}_{i=1}^N$ and $\{\tilde{b}_i\}_{i=1}^N$ by sweeping over l_2 regularisation constants for each member of the ensemble in Listing 3. This step is followed by searching for best fusion weights $\mathbf{w}$ (Eq 8) as shown in Listing 4. We then relearn $\{U\}_{i=1}^N$ and $\{b\}_{i=1}^N$ on combined train and validation data and fuse the

	Dataset	Classes	Original train examples	Train examples	Valid. examples	Test examples	Accuracy measure	Test provided
Classification	CIFAR-10 [36]	10	50000	45000	5000	10000	Top-1 accuracy	-
	CIFAR-100 [36]	100	50000	44933	5067	10000	Top-1 accuracy	-
	Cars [35]	196	8144	6494	1650	8041	Top-1 accuracy	-
	Aircraft [43]	100	3334	3334	3333	3333	Mean per-class accuracy	Yes
	DTD (split 1) [17]	47	1880	1880	1880	1880	Top-1 accuracy	Yes
	Caltech-101 [26]	101	3060	2550	510	6084	Mean per-class accuracy	-
	Flowers [47]	102	1020	1020	1020	6149	Mean per-class accuracy	Yes
Regression	300W [1]	136	599	179	180	240	R2	-
	Leeds Sports Pose [33]	28	1200	960	240	800	R2	-
	CelebA [41]	10	162770	162770	19867	19962	R2	Yes
	Animal Pose [10]	40	6117	3760	1179	1178	PCK@0.05	-
	MPII Human Pose [2]	32	3498	2099	700	699	PCK@0.05	-
	ALOI [29]	1	24000	14400	4800	4800	R2	-
	Causal3D [60]	10	255200	204160	51040	255200	R2	Yes

Table 5. Details of downstream classification and regression datasets used to evaluate QD4V pretraining.

N	CIFAR10	CIFAR100	Flowers	Caltech 101	DTD	Cars	Aircraft	300w	LS Pose	CelebA	Animal Pose	MPII	ALOI	Causal3D	Rank
1 (Baseline)	**90.4**	68.2	85.2	84.7	71.9	44.4	36.0	70.2	55.4	49.0	11.2	18.0	24.1	64.1	4.6
3	90.2	**72.4**	85.7	90.1	72.0	**46.3**	39.6	75.4	57.4	60.8	12.4	18.5	26.7	71.8	2.7
5^*	90.3	70.4	**86.9**	**89.9**	**72.2**	45.4	**36.9**	**76.6**	62.4	**61.5**	**12.5**	18.5	**28.4**	**72.8**	**1.8**
6	90.3	71.3	86.8	89.7	72.0	45.3	**36.9**	76.3	**62.7**	61.3	**12.5**	**18.9**	27.5	71.9	2.0

Table 6. Ablation study over the size of the ensemble. We report the downstream performance of Supervised QD4V pre-training for many-shot classification and regression tasks. 5^* corresponds to results reported in Table 1.

test set predictions of ensemble members with weights $\mathbf{w}$. Finally, we report the corresponding evaluation metric on the test set of the data.

B.1. Classification

For classification, we evaluate on standard benchmarks mentioned in Table 5. We fit a multinomial logistic regression model (from sklearn package) on the extracted features of dimensionality 2048 from the frozen backbones. We do not apply any augmentation and the images were resized to 224 pixels along the shorter side using bicubic resampling, followed by a center crop of 224×224. We select the l_2 regularisation constant on the validation set over 45 logarithmically spaced values between 10^{-6} and 10^5. The model is optimised using L-BFGS on the softmax cross-entropy objective.

B.2. Regression

For regression, we consider a diverse range of tasks with varying invariances or sensitivities. Our aim is to consider a wide range of tasks spanning spatial and appearance sensitivities. We consider common spatially sensitive tasks like

- **Facial landmark prediction:** 300W [1], CelebA [41].
- **Pose estimation:** MPII [2], Leeds Sports Pose [33] and Animal Pose [10].
- **6D pose estimation:** Causal3dIdent.

and appearance sensitive tasks like

- **Object hue prediction:** Causal3DIdent
- **Object orietation prediction:** Learning to predict pose of an obhect based on the variations in lighting conditions in ALOI [29]

We report Percentage of Correct keypoints (PCK with threshold of 0.05) for MPII and Animal Pose, and for the rest we report R2 score. We present more details on dataset splits in Table 5. For regression tasks, we fit a multi-output linear regression model (from sklearn package) on the extracted features of the backbones. Image pre-processing pipelines is similar to classification tasks. We select the $l2$ regularisation constant on the validation set over 100 logarithmically spaced values between 10^{-2} and 10^5. The model is optimised on the Mean Squared Error (MSE) objective.

B.3. Object detection

We train the detector models on the VOC 2007 and 2012 train- val sets, and test on VOC 2007 test. When evaluating frozen backbones, we freeze all the residual blocks of the ResNets. We extract features from the backbone using a Feature Pyramid Network architecture [39] and attach a Faster R-CNN [55] detector head to produce bounding box predictions. Similar to linear/logistic classifiers for regression/classification, a separate Faster-RCNN head is learned for each member of the ensemble. During training, the images are resized so the shorter side is one of [480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800] and during testing to 800 pixels. The models are trained for 144k iterations with a 100 iteration warm-up to an initial learning rate of 0.0025 which is decayed by a factor of 10 at iterations 96k and 128k. The batch size is 2 and we used a single GPU per

N	CUB		Flowers		FC 100		Plant Disease		300w		LS Pose		CelebA		Causal3D		Rank
	(5, 1)	(5, 5)	(5, 1)	(5, 5)	(5, 1)	(5, 5)	(5, 1)	(5, 5)	$s = 0.05$	$s = 0.2$	$s = 0.05$	$s = 0.2$	$s = 0.05$	$s = 0.2$	$s = 0.05$	$s = 0.2$	
1 (Baseline)	70.0±0.5	90.4±0.3	77.3±0.2	93.7±0.3	53.8±0.5	78.7±0.4	68.9±0.3	88.8±0.4	20.5±0.5	24.8±0.3	46.9±0.6	48.5±0.1	39.2±0.5	41.4±0.5	58.9±0.1	62.4±0.7	3.3
3	71.0±0.5	90.5±0.3	73.1±0.5	93.3±0.4	52.9±0.4	75.3±0.4	68.1±0.5	91.0±0.2	35.5±0.1	40.6±0.3	49.9±0.4	50.7±0.2	42.8±0.9	50.9±0.6	59.4±0.2	62.9±0.7	3.0
5*	71.4±0.3	90.8±0.4	73.4±0.2	93.2±0.4	57.5±0.1	76.2±0.6	69.4±0.3	91.0±0.3	37.5±0.4	44.9±0.2	59.7±0.4	60.3±0.2	50.5±0.6	53.1±0.3	60.3±0.4	63.4±0.1	1.9
6	70.5±0.1	91.0±0.3	73.7±0.1	93.6±0.1	57.2±0.3	76.5±0.2	69.2±0.1	91.0±0.3	37.6±0.8	44.7±0.6	59.8±0.6	60.4±0.8	50.8±0.7	53.8±0.1	60.4±0.5	63.5±0.2	**1.5**

Table 7. Ablation study over the size of the ensemble. We report the downstream performance of Supervised QD4V pretraining for few-shot classification and regression tasks. 5^* corresponds to results reported in Table 2.

	Method	DomainNet	CIFAR10 → STL10	Living17	Entity30	ImageNet-R	ImageNet-Sketch	ImageNet-A	Rank
MoCo	LP	79.7 ± 0.6	85.1 ± 0.2	82.2 ± 0.1	**63.2 ± 1.3**	70.6	46.4	45.7	2.5
	FT	55.5 ± 2.2	82.4 ± 0.4	77.8 ± 0.7	55.5 ± 2.2	52.4	40.5	27.8	4
	LP-FT	**80.7 ± 0.9**	**90.7 ± 0.3**	82.6 ± 0.3	62.3 ± 0.9	**72.9**	48.4	49.1	**1.8**
	Ours	74.7 ± 1.1	90.1 ± 0.2	**83.4 ± 0.1**	62.5 ± 0.3	71.6	**50.7**	**49.7**	**1.8**

Table 8. Out of distribution (OOD) accuracies with 90% confidence intervals over 3 runs for DomainNet, CIFAR10 → STL10, Living17, and Entity30. For ImageNet-A, ImageNet-Sketch, and ImageNet-A, we report the average accuracy over 3 runs. QD pre-training performs comparably to LP-FT performance despite using no backpropagation.

model. Any other details of training uses the default values of the detectron2 [67] framework.

B.4. Semantic segmentation

We evaluate QD4V on popular semantic segmentation datasets like Cityscapes and ADE20k datasets. We train a linear layer similar to [4] on the representations obtained from the frozen backbone to produce segmentation masks. For all dataset, the images are of size 512×512. Given the output feature map of dimension $(2048, 32, 32)$ of a ResNet-50, the map is first fed to a linear layer that outputs a map of dimension $(\text{num}_{classes}, 32, 32)$, which is then upsampled using bilinear interpolation to the predicted mask of dimension $(\text{num}_{classes}, 512, 512)$. We use the 40k iterations schedule available in mmsegmentation [18] with both ADE20k and Cityscapes, with the SGD optimizer, and use a learning rate of 0.001.

C. Additional results

C.1. Ablation over the size of the ensemble

In Tables 1, 2, and 3, we present the performance of a QD ensemble consisting of $N = 5$ members. In this section, we explore the effect of varying the ensemble size by performing an ablation study with N ranging from 1 (baseline) to 6. The results for many-shot and few-shot classification and regression are shown in Tables 6 and 7, respectively. As we increase the number of encoders in the ensemble, each member can learn more diverse invariances, which can benefit tasks that require specific sensitivities. Our experiments demonstrate that increasing the number of members from 3 to 5 generally leads to improved performance. However, we observe only marginal performance gains with $N = 6$.

To summarize, our ablation study reveals that increasing the size of the QD ensemble can improve performance on many-shot and few-shot classification and regression tasks. However, there is a diminishing return in performance gains beyond a certain point. The optimal number of members in the ensemble appears to be around 5, which strikes a balance between diversity and complexity.

Model	Brightness	Contrast	Grayscale	Rotation	Flipping
Supervised	0.54	0.64	0.81	0.43	0.92
MoCo	0.77	0.79	0.96	0.45	0.95
CLIP	0.52	0.59	0.71	0.58	0.87

Method	300w	LS Pose	CelebA	Animal Pose	MPII	ALOI	Causal3D
MoCo	87.2	69.0	92.5	13.9	19.9	46.0	78.1
CLIP	87.2	64.2	92.1	12.3	19.2	44.2	75.8
QD (MoCo)	**88.9**	**71.8**	**95.2**	**14.2**	**22.4**	**47.4**	**80.6**

Table 9. **Top:** Comparison of invariances exhibited by ImageNet pretrained, MoCo and CLIP models. **Bottom:** Downstream performance of MoCo, CLIP and QD4V for regression tasks.

C.2. Results on distribution shift datasets

We also evaluate our framework under OOD (Out-of-distribution) conditions for downstream tasks DomainNet [59], CIFAR10 → STL10 [28], ImageNet-sketch[62], ImageNet-A [31], ImageNet-R [31], Living17 and Entity30[56] as proposed by [37].

Does QD4V benefit distribution shift in downstream tasks? We evaluate the effectiveness of QD pre-training on distribution shift datasets and compare it with other methods such as Linear Probing (LP), Finetuning (FT), and LP-FT, which is a two-step strategy of linear probing with full fine-tuning. As shown in [37], fine-tuning can perform worse than linear probing out-of-distribution (OOD) when the pre-trained features are good and the distribution shift is large, while LP-FT leads to performance gains for OOD tasks. In Table 8, we demonstrate that QD4V pre-training achieves comparable performance to LP-FT, while being more computationally efficient as we only need to perform linear readout instead of full fine-tuning. We train a linear classifier on ID train set, search for α^* on ID val test and report performance of the weighted average of ensemble predictions of the OOD task. This highlights the potential of QD pre-training as a robust and efficient method for handling distribution shifts in real-world scenarios.

Can QD4V help large-scale image-text pre-training? CLIP (Contrastive Language–Image Pre-training) by OpenAI [53] bridges vision and language understanding through joint training on text-image pairs. CLIP uses two encoders: a vision encoder based on ViTs or Modified version of ResNet50 for images and a language encoder for text. During training, these encoders learn to associate images with their corresponding textual descriptions through a contrastive objective, optimizing the similarity between correct pairs while minimizing it for incorrect ones. Similar to many contrastive learning works, CLIP is widely evaluated on downstream *classification* tasks, and less on the spatially sensitive tasks that must also be solved by a general purpose feature. Our findings in Table 9 (bottom), reveal that CLIP-RN50 underperforms self-supervised models such as MoCo and QD4V, despite being trained on massively more data. We attribute this disparity to CLIP's high level of invariance to both spatial and appearance- transformations (Table 9 (top)), which is detrimental to pose sensitive regression tasks. In this regard, we believe that incorporating diverse invariances with QD4V pre-training for could benefit vision-language models.

```

class BranchedResNet(nn.Module):
    def __init__(self, N, arch, num_classes, stop_grad = True):
        super(BranchedResNet, self).__init__()

        #  Load ImageNet pretrained weights
        self.base_model = resnet50(weights=ResNet50_Weights.IMAGENET1K_V2)
        self.num_feat = 2048
        self.N = N + 1 # Number of encoders, adding 1 because we add the baseline to ensemble for L_KL
        self.num_classes = num_classes

        del self.base_model.layer4, self.base_model.fc

        # Creating an ensemble by branching out ResNet50 layers
        # Branching out layer 4 and fc of Resnet50, initialising them with pretrained model too, using imagenet pretrained weights only for sipervised pre-training
        self.base_model.branches_layer4 = nn.ModuleList(
                              [resnet50(weights=ResNet50_Weights.IMAGENET1K_V2).layer4
                                        for _ in range(self.N)])
        self.base_model.branches_fc = nn.ModuleList(
                            [resnet50(weights=ResNet50_Weights.IMAGENET1K_V2).fc
                                        for _ in range(self.N)])

        if stop_grad: # Freeze gradients of conv1, layer1, layer2, and layer3
            for name, param in self.base_model.named_parameters():
                if 'layer4' not in name and 'fc' not in name:
                    param.requires_grad = False
                print(name, param.requires_grad)

        # Freezing gradients of baseline model in ensemble
        for name, param in self.base_model.branches_layer4[-1].named_parameters():
            param.requires_grad = False
            print(name, param.requires_grad)

        for name, param in self.base_model.branches_fc[-1].named_parameters():
            param.requires_grad = False
            print(name, param.requires_grad)

    def forward(self, x, reshape = True):
        ''' Input: x: [bs, 3, 224, 224]
        Returns:
        A features and outputs '''
        x = self.base_model.conv1(x)
        x = self.base_model.bn1(x)
        x = self.base_model.relu(x)
        x = self.base_model.maxpool(x)
        x = self.base_model.layer1(x)
        x = self.base_model.layer2(x)
        x = self.base_model.layer3(x)
        feats = [self.base_model.avgpool(self.base_model.branches_layer4[i](x)).view(x.shape[0], -1)
                                                for i in range(self.N)]
        outputs = [self.base_model.branches_fc[i](feats[i]) for i in range(self.N)]

        outputs = torch.cat(outputs).reshape(self.N, -1, self.num_classes)
        feats = torch.cat(feats).reshape(self.N, -1, self.num_feat)
        return outputs, feats
```

Listing 1. Code snippet for Branched ResNet 50.

```python

'''
We show how the QD loss is calculated on one mini-batch during pretraining
'''

'''
N: no. of members in ensemble
kl_coeff: \lambda_kl
div_coeff: \lambda_d
x: unaugmented batch
labels: labels(supervised/self-supervised) labels
x_i: unaugmented image in batch
images: DataLoader returns a list of [T_1(x_i), T_2(x_i), ..., T_5(x_i), x_i]
models: Ensemble of models, returns a list of features and outputs
T: set of augmentations consisting of Random resized crop, Color jitter, Gray scale, Gaussian blur,
    Horizontal flip
'''

'''get_aug_wise_images - Given
           [[T_1(x_1), .. T_5(x_1)],  ...[T_1(x_bs), .. T_5(x_bs)]
           function  rearranges the list
           [[T_1(x_1), ..., T_1(x_bs)], ... ..., [T_5(x_1), T_5(x_bs)], [x_i, ..., x_bs]]
           Stacks the list and  returns a batch of shape (batch_size * (num_augs+1), 3, 224, 224) '''

images = get_aug_wise_images(images, args)


logits, feats = models(images)

# Output of unaugmented images since last batch_size*num_augs logits correspond to unaugmented images
orig_image_logits = logits[:, batch_size*num_augs, :] # (N+1, batch_size, 2048)

# calculating loss for ensemble
ce_loss = [criterion(orig_image_logits[i], labels) for i in range(N)]
ce_loss = sum(ce_loss)

# KL divergence between ensemble and baseline model
# Last element in  orig_image_logits corresponds to output of baseline model
# KL_Divergence returns L_KL with \tau = 6.0
kl_div = KL_Divergence(orig_image_logits[:-1], orig_image_logits[-1], T=6)

# Diversity Loss
# get_similarity_vector returns a matrix (N, 5), where i,j-th element corresponds to invariance of
    memeber i to augmentation j
similarity_matrix = get_similarity_vector(feats[:-1], args)
# Pairwise difference of rows -> distance between phenotypes
diversity = get_pairwise_rowdiff(similarity_matrix).sum()

# Total loss
loss =  (1 - kl_coeff) * ce_loss + div_coeff * diff + kl_coeff * kl_div
```

Listing 2. Pytorch style pseudocode for QD4V pretraining for a single mini-batch

```python

''' We fit $N$ sklearn classifiers for the ensemble by sweeping for best regularisation coefficient '''

from sklearn.linear_model import LogisticRegression

def search_hp(classifier, train_feats, train_labels, val_feats, val_labels, wd_range):
    ''' wd_range is the search space for l2 regularisation constant '''
    best_params = {}
    best_score = 0.0
    for wd in wd_range:
        classifier.set_params(wd) # Set regularisation coefficient as wd
        classifier.fit(train_feats, train_labels)
        val_accuracy =  classifier.score(val_feats, val_labels)
        if val_acc > best_score[k]:
            best_params[str(k)] = wd

    return best_params

def fit_classifier(train_feats, train_labels, num_classes, metric):
    ''' train_feats, val_feats: Features of trai, val dataset for all members of enseble, shape = (N, size_of_dataset, 2048)
    train_labels, val_labels: ground truth labels of entire train, val dataset, shape = (size_of_dataset,)
    num_classes: Number of outputs to predict
    metric: Metric to report
    wd_range: search space of weight decay values  '''
    classifiers = [LogisticRegression(2048, num_classes) for i in range (N)]
    '''Select best ridge hyperparameters for all classifiers '''
    best_params= []
    for n in range(N):
        C = search_hp(classifiers[n], train_feats[n], train_labels, val_feats[n], val_labels, wd_range)
        best_params[n] = C
    return classifiers, best_params

```

Listing 3. Code snippet for downstream training - Finding best model parameters for an ensemble of size N. Here we provide a snippet for classification. For regression tasks we change the classifier to Linear Regression and use the corresponding metric

```python

''' Learn fusion weights $w$  '''
from sklearn.linear_model import LinearRegression

def fusion_weights(val_feats, val_labels, classifiers):
''' val_feats: Features of val dataset for all members of enseble, shape = (N, size_of_dataset, 2048)
    val_labels,: One hot labels of val dataset, shape = (size_of_dataset, num_classes)

    We assume that classifiers are fit on train data with best hyper-parameters'''

    val_preds = []

    for n in range(N):
        preds = classifiers[n].predict(val_feats[n])  # softmax probabilities
        val_preds.append(preds)


    weights = LinearRegression(fit_intercept=False).fit(val_preds, val_labels).coef_
    return weights

```

Listing 4. Code snippet for downstream training - Learning fusion weights $\mathbf{w}$

D. Proofs

D.1. Preliminaries

Our proof technique makes use of the empirical Rademacher complexity [5], defined as

$$\hat{\mathfrak{R}}_{\mathcal{D}}(\mathcal{F}) = \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \epsilon_i f(\mathbf{z}_i) \right],$$

where $\mathcal{D} = \{\mathbf{z}_i\}_{i=1}^m$ is a dataset composed of i.i.d. random variables, $\mathcal{F}$ is a class of functions, and ϵ is a vector of m Rademacher random variables. It is typical for $\mathcal{F}$ is be the composition of some loss function with a class of models, gH; i.e., $\mathcal{F} = l \circ \mathcal{H} = \{l \circ h \, : \, h \in \mathcal{H}\}$. One can use the empirical Rademacher complexity to obtain a bound on the expected value of all $f \in \mathcal{F}$ on new data in terms of the value of the function on the training sample

$$\mathbb{E}[f(\mathbf{z})] \leq \frac{1}{m} \sum_{i=1}^m f(\mathbf{z}_i) + 2\hat{\mathfrak{R}}_{\mathcal{D}}(\mathcal{F}) + 3M \sqrt{\frac{\ln(2/\delta)}{2m}},$$

where M is the maximum value f can take.

We will denote by V the block diagonal matrix where the k-th block along the diagonal is given by the row vector $\mathbf{u}^{(k)}$.

D.2. Proof of Lemma 1

Lemma 1. *Assuming $\|f_{\theta_i}(\mathbf{x})\|_2 \ \leq X$ for all $\mathbf{x} \in \mathcal{X}$, that $l(\cdot, \cdot)$ is a 1-Lipschitz loss function, and for all $\mathbf{w}$ and $\mathbf{u}^{(i)}$ that satisfy the constraints in the definition of $\mathcal{H}$, we have that*

$$\hat{\mathfrak{R}}_{\mathcal{D}_{ds}}(l \circ \mathcal{H}) \leq \frac{3ABX + BX\|\mathbf{w}\|_2 + AX\|V\|_2}{\sqrt{m}}.$$

Proof. Let the vector $\mathbf{z}_i$ be the concatenation of the output of each $f_{\theta_k}(\mathbf{x}_i)$, and let $\mathbf{c}$ be a vector such that $c_k = b^{(k)}$. Expanding definitions and leveraging linearity yields

$$\begin{aligned} \hat{\mathfrak{R}}_{\mathcal{D}_{ds}}(l \circ \mathcal{H}) &= \mathbb{E}_\epsilon \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \epsilon_i l(h(\mathbf{x}_i), y_i) \right] \\ &\leq \mathbb{E}_\epsilon \left[\sup_{\mathbf{w}, V} \frac{1}{m} \sum_{i=1}^m \epsilon_i \langle \mathbf{w}, V\mathbf{z}_i \rangle \right] \\ &= \mathbb{E}_\epsilon \left[\sup_{\mathbf{w}, V} \frac{\mathbf{w}^T V}{m} \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right], \end{aligned}$$

where the inequality comes from the contraction inequality for Rademacher complexities. Expanding the definitions of $\mathbf{w}$ and V gives us

$$\mathbb{E}_\epsilon \left[\sup_{\Delta \mathbf{w}, \Delta V} \frac{(\mathbb{E}[\mathbf{w}^T] + \Delta \mathbf{w}^T)(\mathbb{E}[V] + \Delta V)}{m} \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right],$$

where $\Delta \mathbf{w} = \mathbf{w} - \mathbb{E}[\mathbf{w}]$ and $\Delta V = V - \mathbb{E}[V]$. We can then expand this into

$$\mathbb{E}_\epsilon \left[\sup_{\Delta \mathbf{w}, \Delta V} \frac{\mathbb{E}[\mathbf{w}^T]\mathbb{E}[V] + \mathbb{E}[\mathbf{w}^T]\Delta V + \Delta \mathbf{w}^T \mathbb{E}[V] + \Delta \mathbf{w}^T \Delta V}{m} \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right].$$

From the distributivity of inner products, we can separate out the $\mathbb{E}[w^T]\mathbb{E}[V]$ term into a new Rademacher complexity term. This new term evaluate to zero because there are no learnable parameters in it. Subsequently, we can liberally apply the Cauchy-Schwarz and triangle inequalities to obtain

$$\mathbb{E}_\epsilon \left[\sup_{\Delta \mathbf{w}, \Delta V} \frac{\|\mathbb{E}[\mathbf{w}^T]\|_2 \|\Delta V\|_2 + \|\Delta \mathbf{w}^T\|_2 \|\mathbb{E}[V]\|_2 + \|\Delta \mathbf{w}^T\|_2 \|\Delta V\|_2}{m} \left\| \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right\|_2 \right].$$

Now note that for any $\mathbf{w}$ that satisfies $\|\mathbb{E}[\mathbf{w}] - \mathbf{w}\|_2 \leq A$ we have that $\|\mathbb{E}[\mathbf{w}]\|_2 \leq \|\mathbf{w}\|_2 + A$ (and similar for V), allowing us to further bound the complexity by

$$\mathbb{E}_\epsilon \left[\frac{(\|\mathbf{w}\|_2 + A)B + A(\|V\|_2 + B) + AB}{m} \left\| \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right\|_2 \right].$$

This can then be simplified to

$$\mathbb{E}_\epsilon \left[\frac{3AB + B\|\mathbf{w}\|_2 + A\|V\|_2}{m} \left\| \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right\|_2 \right].$$

Using a standard techniques (see, e.g., the proof of Lemma 26.10 in [57]), and that $\|\mathbf{z}_i\|_2 \leq X$, we have that

$$\mathbb{E}_\epsilon \left[\left\| \sum_{i=1}^m \epsilon_i \mathbf{z}_i \right\|_2 \right] \leq \sqrt{m} X.$$

Applying this bound yields

$$\frac{3ABX + BX\|\mathbf{w}\|_2 + AX\|V\|_2}{\sqrt{m}}.$$

$\square$

D.3. Proof of Lemma 2

Lemma 2. *For* $\mathbf{w}$ *and* V *obtained using the procedure in Section 3.3, where the loss function is chosen to be the hinge,* $l(t, y) = \max(0, 1 - ty)$*, we have with probability at least* $1 - \delta$ *that*

$$\|\mathbf{w} - \mathbb{E}[\mathbf{w}]\|_2 \leq \frac{\|\tilde{V}\|_2 X}{\gamma_2} \sqrt{\frac{2 \ln(4/\delta)}{(1-\eta)m}}$$

and

$$\|V - \mathbb{E}[V]\|_2 \leq \frac{X}{\gamma_1} \sqrt{\frac{2 \ln(4N/\delta)}{m}},$$

where the inputs, $\mathbf{x}$*, to the first layer of linear models satisfy* $\|\mathbf{x}\| \leq X$*,* γ_1 *is the* ℓ_2 *regularisation hyperparameter for the first layer, and* γ_2 *is the hyperparameter for the second layer.*

In the course of proving this Lemma we will use the following results due to [40] and [64].

Lemma 3 (Specialisation of Lemma 1 in [40] for Hilbert spaces)**.** *Let A be an algorithm that maps a training set, $\mathcal{D}$, of size m to some model parameters in a Hilbert space. Then with probability at least $1-\delta$*

$$\|A(\mathcal{D}) - \mathbb{E}[A(\mathcal{D})]\|_2 \leq \alpha(m)\sqrt{2m\ln(2/\delta)},$$

where $\alpha(\cdot)$ is the uniform argument stability of A.

Lemma 4 (Proposition 3 in [40], implied by Theorem 3.5 in [64])**.** *For ℓ_2 Regularised empirical risk minimisation of linear models with L-Lipschitz and convex loss functions bounded by M, we have that*

$$\alpha(m) \leq \frac{XL}{\gamma m},$$

where the inputs, $\mathbf{x}$, to the linear model satisfy $\|\mathbf{x}\| \leq X$.

We now proceed with the proof of Lemma 2.

Proof. From Lemmas 3 and 4, we have for each $\mathbf{u}_i$ that with probability $1-\delta$

$$\|\mathbf{u}_i - \mathbb{E}[\mathbf{u}_i]\|_2 \leq \frac{X}{\gamma}\sqrt{\frac{2\ln(2/\delta)}{m}}. \tag{11}$$

Now note that the spectral norm of a block diagonal matrix is the maximum of the spectral norms of each block. We can combine the bounds on each $\mathbf{u}_i$ into a bound on the deviation of V from its expected value using this fact and the union bound; with probability $1-\delta$,

$$\|V - \mathbb{E}[V]\|_2 \leq \frac{X}{\gamma}\sqrt{\frac{2\ln(2N/\delta)}{m}}. \tag{12}$$

To get a bound on the deviation of $\mathbf{w}$ from its expected value, note that we can just reuse Equation 11 with $\mathbf{w}$ in place of $\mathbf{u}_i$, and that instead of an upper bound on $\|\mathbf{x}\|_2$, we require an upper bound on $\|\tilde{V}\mathbf{x}\|_2$. Such a bound arises from the definition of the operator norm,

$$\|\tilde{V}\mathbf{x}\|_2 \leq \|\tilde{V}\|_2 X.$$

Finally, the 2 inside the logarithm of each bound becomes a 4 due to the union bound, thus ensuring both bounds in the Lemma statement will hold simultaneously with probability at least $1-\delta$. □

D.4. Proof of Theorem 1

Theorem 1. *For a model trained using our stacking procedure and the conditions outlined in the statements of Lemmas 1 and 2, we have with probability at least $1-2\delta$,*

$$\begin{aligned}\mathbb{E}[\mathbf{1}[\mathrm{sgn}(h(\mathbf{x})) \neq y]] &\leq \hat{\mathbb{E}}[l(h(\mathbf{x}), y)] \\ &\quad + \frac{12X^3\|\tilde{V}\|_2\sqrt{\ln(4/\delta)\ln(4N/\delta)}}{\gamma_1\gamma_2\sqrt{1-\eta}m^{3/2}} + \frac{2X^2\sqrt{2\ln(4N/\delta)}}{\gamma_1 m} + \frac{2X^2\|\tilde{V}\|_2\sqrt{2\ln(4/\delta)}}{\gamma_2\sqrt{1-\eta}m} \\ &\quad + 3\sqrt{\frac{\ln(1/\delta)}{2m}},\end{aligned}$$

where $\hat{\mathbb{E}}[l(h(\mathbf{x}), y)]$ is the mean ramp loss of the model h, computed on the training data.

Proof. First, note that the ramp loss is an upper bound for the zero–one loss, allowing us to use the ramp loss for the empirical error term in the bound on the zero–one loss for the expected error. From Lemmas 1 and 2 we have with confidence at least $1-\delta$ that

$$\begin{aligned}\hat{\mathfrak{R}}_{\mathcal{D}_{ds}}(l \circ \mathcal{H}) &\leq \frac{3ABX + BX\|\mathbf{w}\|_2 + AX\|V\|_2}{\sqrt{m}} \\ &= \frac{6X^3\|\tilde{V}\|_2\sqrt{\ln(4/\delta)\ln(4N/\delta)}}{\gamma_1\gamma_2\sqrt{1-\eta}m^{3/2}} + \frac{X^2\sqrt{2\ln(4N/\delta)}}{\gamma_1 m} + \frac{X^2\|\tilde{V}\|_2\sqrt{2\ln(4/\delta)}}{\gamma_2\sqrt{1-\eta}m}.\end{aligned}$$

Merging this, via the union bound, with the standard Rademacher complexity generalisation bound [5] completes the proof. □

References

[1] 300 faces in-the-wild challenge: database and results. *Image and Vision Computing*, 2016. 300-W, the First Automatic Facial Landmark Detection in-the-Wild Challenge. 5, 13

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 5, 13

[3] Arsenii Ashukha, Andrei Atanov, and Dmitry Vetrov. Mean embeddings with test-time data augmentation for ensembling of representations. *ICML Workshop on Uncertainty and Robust- ness in Deep Learning*, 2021. 6, 7

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicregl: Self-supervised learning of local visual features. *NIPS*, 2022. 5, 14

[5] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. 19, 21

[6] Gregory W Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks. In *NIPS*, 2020. 2

[7] Xavier Puig Sanja Fidler Adela Barriuso Bolei Zhou, Hang Zhao and Antonio Torralba. The amsterdam library of object images. *CVPR*, 2017. 5, 8

[8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *ArXiv*, 2021. 2

[9] Aleksander Botev, Matthias Bauer, and Soham De. Regularising for invariance to data augmentation improves supervised learning. *arXiv*, 2022. 2

[10] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation. In *ICCV*, 2019. 5, 13

[11] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NIPS*, 2020. 1

[12] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. 2021. 2

[13] Ruchika Chavhan, Jan Stuehmer, Calum Heggan, Mehrdad Yaghoobi, and Timothy Hospedales. Amortised invariance learning for contrastive self-supervision. In *ICLR*, 2023. 1, 2, 6, 7

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 3

[15] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *ArXiv*, 2020. 1, 12

[16] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *ICCV*, 2021. 3

[17] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 5, 13

[18] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 7, 14

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 8

[20] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 2015. 2, 3

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[22] Thomas Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2000. 3

[23] N. Dvornik, J. Mairal, and C. Schmid. Diversity with cooperation: Ensemble methods for few-shot classification. In *ICCV*, 2019. 6, 7, 12

[24] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *CVPR*, 2021. 1

[25] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. Why do self-supervised models transfer? investigating the impact of invariance on downstream tasks. *BMVC*, 2022. 1, 2, 3, 6

[26] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004. 5, 13

[27] Matthew Christopher Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. In *NIPS*, 2021. 2, 3

[28] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *ICLR*, 2018. 14

[29] Burghouts G.J. Smeulders A.W. Geusebroek, JM. The amsterdam library of object images. *IJCV*, 2005. 5, 13

[30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 3

[31] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 14

[32] Alexander Immer, Tycho FA van der Ouderaa, Vincent Fortuin, Gunnar Rätsch, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations. *NIPS*, 2022. 2

[33] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*. Aberystwyth, UK, 2010. 5, 13

[34] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019. 1

[35] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop on 3D Representation and Recognition (3dRR-13)*, 2013. 5, 13

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. 5, 13

[37] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022. 14

[38] Hankook Lee, Kibok Lee, Kimin Lee, Honglak Lee, and Jinwoo Shin. Improving transferability of representations via augmentation-aware self-supervision. *NIPS*, 2021. 2, 6, 7, 8, 12

[39] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *CVPR*, 2016. 5, 13

[40] Tongliang Liu, Gábor Lugosi, Gergely Neu, and Dacheng Tao. Algorithmic Stability and Hypothesis Complexity. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2159–2167. PMLR, July 2017. 21

[41] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 5, 13

[42] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 5

[43] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 5, 13

[44] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 2016. 7

[45] João Moreira, Carlos Soares, Alípio Jorge, and Jorge Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys*, 2012. 3, 4

[46] Ury Naftaly, Nathan Intrator, and David Horn. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 1997. 3

[47] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICCVGIP*, 2008. 5, 13

[48] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *Arxiv*, 2018. 3

[49] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *NIPS*, 2018. 7

[50] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *ICML*, 2019. 7, 12

[51] Justin Pugh, Lisa Soros, and Kenneth Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 2016. 2, 3

[52] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *NIPS*, 2020. 2

[53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 8, 15

[54] Aniruddh Raghu, Jonathan Lorraine, Simon Kornblith, Matthew McDermott, and David K Duvenaud. *NIPS*, 2021. 2

[55] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 5, 13

[56] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. In *ICLR 2021*, 2020. 14

[57] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 20

[58] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *ICANN*, 2018. 1

[59] Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey. In Adrien Bartoli and Andrea Fusiello, editors, *ECCV*, 2020. 14

[60] Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. *NIPS*, 2021. 5, 13

[61] Diane Wagner, Fabio Ferreira, Danny Stoll, Robin Tibor Schirrmeister, Samuel Müller, and Frank Hutter. On the importance of hyperparameters and data augmentation for self-supervised learning. *ICML Pre-training Workshop*, 2022. 2

[62] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NIPS*. Curran Associates, Inc., 2019. 14

[63] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *ICML*, 2020. 2

[64] Andre Wibisono, Lorenzo Rosasco, and Tomaso Poggio. Sufficient Conditions for Uniform Stability of Regularization Algorithms. Technical report, MIT, Dec. 2009. 21

[65] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 5

[66] David H. Wolpert. Stacked generalization. *Neural Networks*, 1992. 2, 4

[67] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 7, 14

[68] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2021. 1, 2, 3, 4, 5, 7, 8

[69] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021. 1