Spatial Mixture-of-Experts

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many data have an underlying dependence on spatial location; it may be weather on the Earth, a simulation on a mesh, or a registered image. Yet this feature is rarely taken advantage of, and violates common assumptions made by many neural network layers, such as translation equivariance. Further, many works that do incorporate locality fail to capture fine-grained structure. To address this, we introduce the Spatial Mixture-of-Experts (SMoE) layer, a sparsely-gated layer that learns spatial structure in the input domain and routes experts at a fine-grained level to utilize it. We also develop new techniques to train SMoEs, including a self-supervised routing loss and damping expert errors. Finally, we show strong results for SMoEs on numerous tasks, and set new state-of-the-art results for medium-range weather prediction.

1 Introduction

Many datasets exhibit an underlying, location-based structure, where the value at a point depends on where that point is. For example, weather predictions [67] depend on their location on Earth; many scientific simulations are parameterized on an underlying mesh [8]; data (e.g., faces) may be specifically aligned [80]; or it may approximately hold, as in natural images with centered objects. Further, tasks on such data are often dense regression tasks, such as predicting weather several days in the future. Numerous architectures have been successfully applied to such tasks. Convolutional neural networks (CNNs) [69] and transformers [63] show promise for medium-range weather prediction. Locally-connected networks (LCNs), which use independent, rather than shared, filters at each point, have been applied to weather post-processing [31], face recognition [37, 66, 80], and other tasks [13, 60], specifically to learn local features. Low-rank local connectivity (LRLCN) [25] relaxes the translation equivariance of CNNs while requiring fewer parameters than LCNs. Other approaches, such as CoordConv [54], add an additional inductive bias by providing explicit input coordinates.

However, for tasks on data with location-based structure, prior approaches suffer from various limitations. Convolution assumes that data is translation equivariant, which does not hold for such tasks [37, 80]. Approaches like LCNs require many parameters. Many architectures have been designed for classification tasks and fail to perform well on regression because they opererate at too coarse granularity and are unable to capture key details. This limits their applicability to important tasks, such as medium-range weather prediction or climate simulations. Indeed, on a simple heat diffusion task with location-dependent diffusivities (see §3.1), many approaches do not learn the location dependence at all and instead converge to an "average" diffusivity.

To address this, we introduce a novel neural network layer, the Spatial Mixture-of-Experts (SMoE) layer (§2). An SMoE uses a learned gating function to sparsely select and route from a shared set of experts to each spatial location (e.g., pixel) in an input sample. This enables experts to specialize to the unique characteristics of different "regions" within the data and easy interpretability by analyzing the gate. SMoEs require the assumption that all samples in the dataset have a similar underlying spatial structure, but this often holds (at least approximately) for many datasets, such as weather,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

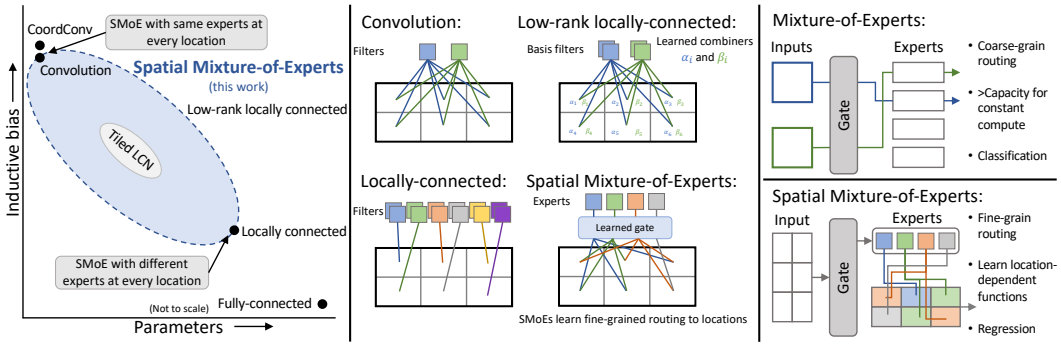

Figure 1: Comparison of SMoEs to other networks. Left: Qualitative comparison of parameter counts and inductive biases. Center: Pattern of filter application. Right: Routing in MoEs and SMoEs.

where each example is on the same latitude/longitude grid. For the SMoE gating function, we introduce *tensor routing*, a simple and cheap routing function that effectively learns this spatial structure (§2.1). We also find that end-to-end training approaches, as in standard mixture-of-experts (MoEs) [71, 75], do not capture location-dependence well. Instead, we introduce a separate *routing classification loss* (§2.2) to train the gate, a self-supervised loss which provides a better learning signal to the gate about incorrect routings. We also introduce *expert error damping* (§2.3), which limits expert updates from misroutings, and further improves performance. Both methods rely on extracting information from the error signal (gradient of the layer output) which would otherwise be uninformative. Figure 1 provides an overview of SMoEs and a qualitative comparison to important related work. With these methods, SMoEs are able to learn fine-grain, location-dependent which other architectures miss, and consequentially deliver much better performance.

SMoEs are related to and inspired by prior work in MoEs (e.g., [71, 75, 88]), but there are key differences. Existing MoEs use coarse-grained routing at the sample [86], token [75], or patch [71] level, and experts produce coarse output (e.g., entire samples or channels) while SMoEs route at a fine-grained, per-pixel level. Fine-grained routing is important for enabling experts to specialize to fine-scale information, which may be crucial [63, 73]. Such MoEs also typically aim to increase model capacity while keeping compute costs constant, whereas the goal of SMoEs is to capture spatial dependencies. Other work has aimed to incorporate spatial dependence, such as LRLCNs [25], which learn content- and location-dependent weights to combine a set of basis filters. However, we found that these prior methods failed to capture the fine-grained features necessary for good performance on regression tasks (§3.1).

We conduct experiments on several benchmark datasets with SMoEs and conduct extensive ablation studies of SMoE design decisions (§3). Using a simple heat diffusion dataset, we showcase the limitations of other models and the power of SMoEs in a controlled situation. We then apply SMoEs to medium-range weather prediction and outperform the state-of-the-art [69] on WeatherBench [67] when not using additional data. Finally, we show that SMoEs can also be applied to image classification tasks, where we match or outperform LRLCNs while using fewer parameters.

1.1 Related Work

Mixture-of-Experts and conditional computation. While MoEs have long been used [12, 40, 43], since the advent of deep learning, there has been much work in applying MoEs, conditional computation, and dynamic routing to DNNs [1, 4, 5, 7, 9–11, 14–16, 18, 20, 22, 24, 27, 28, 32, 42, 49, 50, 59, 64, 65, 71, 72, 75, 86, 88, 91]. Often, the goal is to increase model capacity without a corresponding increase in compute costs and use coarse-grained routing. Notably, MoEs route at the sample (e.g., [20, 24, 59, 72]), token (e.g., [27, 49, 75]), or patch (e.g., [71]) level; are typically trained with extensive auxiliary losses; and often target classification tasks. This coarse routing, in particular, means there is limited opportunity for experts to specialize. MoEs specifically for vision tasks also typically operate at a sample-level (e.g., [1, 4, 32, 86, 88]) and use experts to specialize filter or channel choices. In contrast, our SMoEs induce an inductive bias via fine-grained, location-

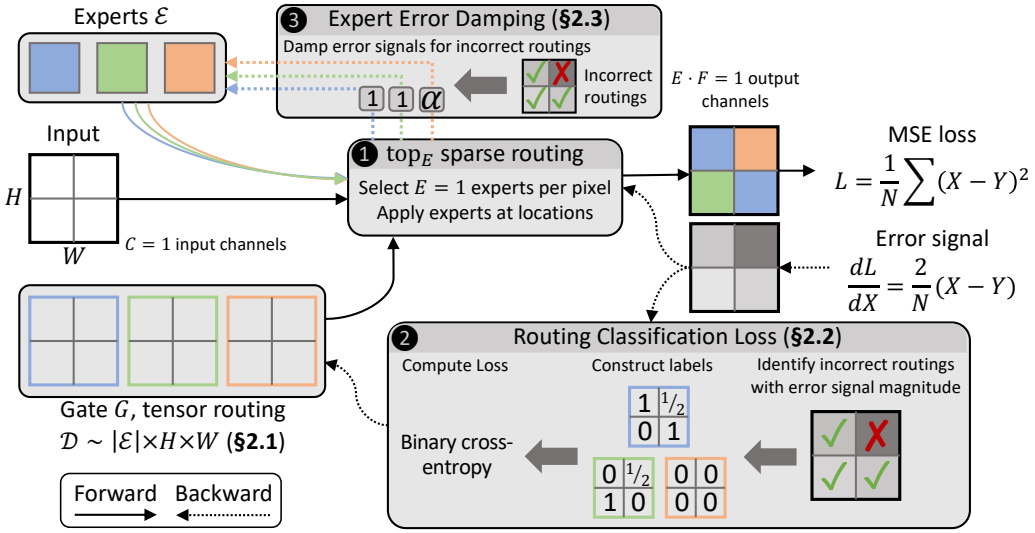

Figure 2: Overview of SMoE architecture and training using a mean-square error loss and an example input. ❶ Each location in an input is routed to E experts from a set $\mathcal{E}$ based on a gate G. ❷ The gate is trained using a routing classification loss, which identifies incorrect routings based on the error signal. ❸ The error signal to experts is damped for locations they were incorrectly routed to.

dependent routing for problems with such underlying structure; typically do not need auxiliary losses beyond the routing classification loss; and work well for dense regression tasks.

Local spatial structure. Locally connected networks (i.e., convolution filters without weight sharing) were historically successful in vision tasks such as facial recognition [13, 37, 66, 80]. Further work has incorporated periodic or tiled weight sharing [30, 60, 90]. However, more recent work often exclusively uses convolution, which has been found to perform better [61] while requiring significantly fewer parameters. CoordConv [54] explicitly provides pixel coordinates as input. Because of their structure, a single convolution or CoordConv layer is unable to learn the location-dependent sparse activations as SMoEs do, and multi-layer networks give limited improvements in practice. Low-rank locally connected networks [25] learn to combine a small number of basis filters in a location- and input-dependent manner. These combiners lack sparsity and apply all basis filters at each point, and are softmax-normalized, which can result in a few filters dominating, limiting diversity. Other work has used attention to provide position-dependent scaling or to integrate context [29, 41, 53, 81, 83, 87]. This has culminated in vision transformers [21] and variants [23, 51, 55, 84, 85], which use specialized attention mechanisms to integrate spatial information.

Sparsity. Our use of sparse routing resembles work on sparsity in DNNs in general [36]. Many approaches learn a fine-grained mask to identify which weights to keep or prune (e.g., [52, 79]), while other approaches sparsify activations (e.g., [3, 38, 46, 57, 58]). These works use sparsity to improve performance, typically at inference time.

Weather prediction. Medium-range weather prediction, or forecasting roughly three to seven days in advance [6], is of broad societal import [47]. There has been much interest in applying deep learning to this task [74], and WeatherBench [67] serves as a community benchmark. Deep learning has also been successfully applied to the related tasks of ensemble post-processing [31, 68] and now-casting (forecasting a few hours in advance) [2, 26, 70, 76–78]. Standard CNNs are currently state-of-the-art on WeatherBench [69]. However, weather data has extensive fine-grained, location-dependent structure which take advantage of with SMoEs for improved performance.

2 Spatial Mixture of Experts

We now introduce the Spatial Mixture-of-Experts (SMoE) layer. An SMoE uses a learned gating function (§2.1) to select specific experts from a shared set to apply at each point (e.g., pixel) in an input. The gate learns the underlying spatial structure of the data, and routes specific experts to

different "regions", allowing them to specialize. An SMoE is predicated on the assumption that the spatial structure is similar across all samples in a dataset. We also assume data are on a Cartesian mesh (i.e., grid), although this could be generalized. To train SMoEs, we introduce a self-supervised *routing classification loss* (§2.2) and *expert error damping* (§2.3), which we found essential for achieving the best performance. Figure 2 provides an overview of SMoEs and their training.

We first define the SMoE layer in full generality, then discuss the particular implementation we use. (See §3.1 for ablations.) Let $x \in \mathbb{R}^{C \times H \times W}$ be an input sample (we assume 2D data for simplicity). The SMoE layer consists of a set $\mathcal{E}$ of experts, of which $E \leq |\mathcal{E}|$ will be selected at each point, and a gating function $G : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{|\mathcal{E}| \times H \times W}$. The gating function has exactly E nonzeros at each spatial point ($H \times W$), which correspond to the selected experts at that point. Each expert $e \in \mathcal{E}$ is applied at the points in x where it has been selected, and may include additional context from x (e.g., surrounding points if e is convolution), and produces an output of size F. The outputs of each expert at a point are then concatenated to form the output channel dimension (of size $E \cdot F$) of the SMoE. More precisely, let $\mathrm{gather}(\cdot)$ select only the non-zero entries of its input. Then an SMoE is:

$$y = \mathrm{gather}\left(G(x) \cdot [e_1(x); \ldots; e_{|\mathcal{E}|}(x)]\right),$$

where $\cdot$ is element-wise multiplication, $[e; \ldots]$ stacks tensors, and $y \in \mathbb{R}^{EF \times H \times W}$. When $E \ll |\mathcal{E}|$, the gating function induces significant sparsity; this allows an SMoE to compute experts only at the points where they are to be applied, avoiding most computation.

This formulation yields a *weighted* SMoE, where the expert outputs are scaled by the gating function. An alternative, which is slightly more efficient and can be more readily interpretable, is an *unweighted* SMoE, in which case experts are not scaled, and the gather only uses $G(x)$ to select experts.

In this work, we focus on a simple, yet powerful, SMoE layer using convolution filters as experts:

$$y_i = G(x)_i \cdot \sum_{c \in [C]} w_{i,c} \star x_c,$$

where $\star$ is cross-correlation, $i \in [|\mathcal{E}|]$, and we have elided the gather for simplicity.

2.1 Gating Functions

The gating function G in an SMoE is critical for learning the underlying location-dependence of the data. While G can be any function, a good gating should be relatively cheap. We follow prior work on MoEs (e.g., [71, 75]) and use top-E routing to select experts and sparsify the gate: $G(x) = \mathrm{top}_E(g(x))$, where $g(x) \in \mathbb{R}^{|\mathcal{E}| \times H \times W}$ is a learnable gating layer and top_E selects the E largest entries along the first (expert) dimension and sets the remaining entries to 0. However, we found that normalizing the gating function using softmax, as is typically done, was not necessary.

There are many options for the gating layer. Many MoE works have used MLPs [71, 75], but we found this did not perform well. Convolution or CoordConv [54] also did not perform well. Instead, we find that a simple *tensor routing* gating layer worked best. With this, $g(x) = \mathcal{D} \in \mathbb{R}^{|\mathcal{E}| \times H \times W}$, where $\mathcal{D}$ is a learnable tensor that directly encodes the underlying location dependence and routing structure without depending on the input. Further, $\mathcal{D}$ uses one parameter per expert per location and requires no computation beyond optimizer updates, making it an efficient choice.

We initialize $\mathcal{D}$ using a uniform distribution over $[-{}^{3|\mathcal{E}|}/_{EF}, {}^{3|\mathcal{E}|}/_{EF}]$, which corresponds to a Kaiming uniform initialization with fan-in ${}^{EF}/_{|\mathcal{E}|}$ [33]. However, in many practical cases, some data about locations may be known (e.g., whether it is land or sea when doing weather prediction). In such cases, $\mathcal{D}$ can be initialized based on this data to assign groups of experts in advance. This allows the gating function to benefit from prior knowledge while still being able to adjust the routing. Additionally, a network may contain many SMoE layers, in which case we can share $\mathcal{D}$ between layers that have the same spatial dimensions and number of experts, which reduces the overhead of the gate.

Finally, many MoE formulations use a number of auxiliary losses to ensure gates learn good routing policies (e.g., [9, 71, 75]). When training with the routing classification loss we propose, we did not find additional auxiliary losses to be necessary.

2.2 Training and the Routing Classification Loss

Training MoEs is typically done end-to-end, with the experts and gate learning simultaneously, and sparse gradients based on the routing. We found this did not lead to good performance with SMoEs and tensor routing, particularly on regression tasks, as the gate did not learn well. Indeed, in such cases, the gate rarely changed its routing decisions from initialization. We hypothesize that this is due to a "mismatch" in the gradients on regression tasks, where they are informative for experts but not the gate (see Appendix B). To address this, we train the gate with a separate, self-supervised loss function, the *routing classification* (RC) loss. The key idea is to consider routing as a dense, multi-label classification task: selecting the "correct" experts at each point (cf. semantic segmentation). The RC loss does exactly this, and trains the gate by constructing appropriate labels.

In order to construct these labels, we need to determine whether the gate selected the correct set of experts at each point. However, such information is not directly available. Instead, we use the error signal into the SMoE layer (i.e., the gradient of the layer output w.r.t. the loss) as a proxy, and say that the routing was incorrect when the error signal has a large magnitude for the expert at that point, as this will imply a correspondingly large gradient update. Further, in the case of a mean-square error loss L (commonly used for regression), the error signal can directly encode the prediction error. Let X be the predictions, Y the true values, and N the number of elements in X. Then the error signal is:

$$\frac{dL}{dX} = \frac{d}{dX}\frac{1}{N}\sum(X-Y)^2 = \frac{2}{N}(X-Y).$$

Hence, the error signal is simply the (scaled, signed) error of the predictions. While this exact relation ceases to hold as backpropagation proceeds, the intuition behind the error signal magnitude remains.

We now define the routing classification loss. Given the error signal ε into an SMoE layer, and an error quantile q (a hyperparameter), we say that selecting an expert at a point was incorrect if ε at that expert and point is greater than the qth quantile of ε, and correct otherwise. We use quantiles as they are independent of ε's scale, which may be hard to determine and change during training. We then construct the labels for each point as follows: Unselected experts start with label 0. A correctly selected expert has label 1. Finally, if an expert was incorrectly selected, its label is 0 and we add $1/(|\mathcal{E}| - E)$ to the label value of each unselected expert (note $|\mathcal{E}| - E$ is the number of unselected experts). This corresponds to a uniform prior that the correct expert could be any unselected expert. With these labels, the RC loss for a gate is then the binary cross-entropy loss of the gate output.

2.3 Expert Error Damping

While the RC loss enables an SMoE gate to be trained directly based on whether it routed correctly, experts that were incorrectly routed to still perform gradient updates based on this routing. This results in experts updating to improve their performance for locations they may not be applied at in future training iterations. To mitigate this, we propose *expert error damping*, where the portion of an error signal that corresponds to incorrect routings is damped to limit incorrect updates. We find that this can improve SMoE performance and reduce convergence time.

Expert error damping is similar to the RC loss, and we classify incorrect routings in the same way. We then scale the error signal into the experts by a constant factor at each point where the routing was incorrect. This will limit the magnitude of the update made in response to the misrouting.

2.4 Practical Implementation

We now discuss the implementation of an SMoE layer, primarily focusing on the simple SMoE with convolution filters we use. In an ideal implementation, the flop cost of an SMoE is the cost of the gating function plus the cost of applying the selected experts. When using a tensor routing gating layer, there are no flops in the gating layer. The flops from applying the selected convolutional experts is equivalent to a standard convolutional layer with a number of filters equal to the number of selected experts. Hence, SMoEs are quite efficient flop-wise. However, recent work has shown that data movement costs are also critical to performance [39]. Because of this, we do not expect even well-optimized implementations to match the runtime of a convolution layer due to the sparse routing, limited locality in accessing expert filters, and other operations, although work on hardware-accelerated sparsity [17] should offer benefits. Despite this, during inference some

Model	#Params	% within 1%
Convolution	146	$91.3^{\pm 0.2}$
CoordConv [54]	56	$91.1^{\pm 0.2}$
CondConv [88]	200	$91.2^{\pm 0.3}$
LRLCN [25]	516	$91.8^{\pm 0.5}$
LRLCN-ND [25]	12315	$91.5^{\pm 0.3}$
LCN	36764	$\mathbf{100.0}^{\pm 0}$
FC	16.7M	$14.2^{\pm 1.2}$
ViT [21]	200k	$93.5^{\pm 0.1}$
V-MoE [71]	470k	$93.8^{\pm 0.3}$
SMoE	27+12288	$\mathbf{100.0}^{\pm 0}$

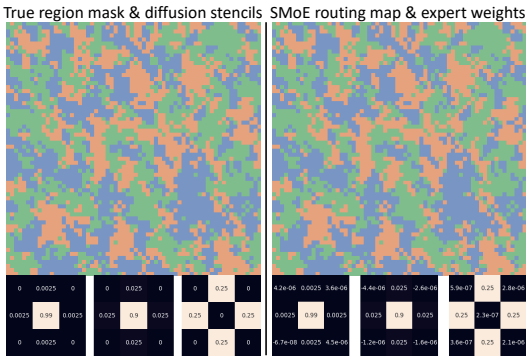

Figure 3: Results on the heat diffusion dataset. Left: Performance of SMoEs and baseline models. Center: Region map and diffusion stencils for the dataset. Right: Learned SMoE routing map and expert weights. SMoEs learn the correct stencils and location-dependence.

additional optimizations may be available because the tensor routing gate does not depend on the input and computations could be reordered to maximize locality.

Unfortunately, efficiently implementing the irregular access patterns in an SMoE is challenging in standard Python deep learning frameworks, and likely requires custom compute kernels to be written. Instead we opt for a naïve implementation in PyTorch [62] where we apply all experts at all points and then use `gather` and `scatter` operations to efficiently implement sparse routing. We find this implementation sufficient even for large-scale tasks.

3 Experiments

We now present experimental results using SMoEs and a variety of baseline models, as well as ablations of different SMoE components. First, we describe a simple location-dependent heat diffusion dataset, which we use to study the ability of different architectures to learn location dependence in a controlled environment (§3.1). We then present results on the WeatherBench medium-range weather forecasting challenge, where SMoEs achieve state-of-the-art accuracy (§3.2). Lastly, we show results on several image classification tasks, illustrating the breadth of SMoE applicability even in situations without strict location-dependence (§3.3).

All results were run using PyTorch [62] version 1.11 on a large cluster with 16 GB V100 GPUs. We summarize training details throughout this section, and provide full details in Appendix A. We use existing PyTorch implementations when available, and implemented methods ourselves otherwise. Unless noted, all SMoE results used 3×3 convolution kernel experts, unweighted tensor routing gates, RC loss, expert error damping, error quantile $q = 0.7$, and damping factor 0.1. We report the mean and standard deviation ($\text{mean}^{\pm \text{std}}$) over ten runs for all experiments, except for WeatherBench and ImageNet, which used three runs. In total, we used about 24k GPU hours for experiments.

3.1 Location-Dependent Heat Diffusion

To study location-dependence in a controlled manner, we generated a heat diffusion dataset with location-dependent diffusivities. The task is, given the current heat at each point, to predict what the heat will be at the next timestep. Because we know the exact diffusivities and their distribution in the data, it is easy to identify how well a model has learned this task. We first describe the dataset and its generation in detail, then discuss results and ablation studies on SMoE components.

Dataset. The dataset consists of a region map, where each location is assigned a type, and each type corresponds to a different heat diffusivity. The region map and diffusivity assignment are then fixed for the dataset. We generate the region map randomly using a simple preferential attachment method. This defines the location-dependence that we wish to learn. To generate samples, we first randomly distribute drops of heat across the domain, then apply five-point diffusion stencils at each point, using the diffusivity of the region type for each point. For simplicity, we use zero boundary conditions.

Table 1: Effect of RC loss and expert error damping on the heat diffusion dataset.

RC loss	Damping	Epochs	% within 1%
✗	✗	14	$91.5^{\pm 0.4}$
✗	✓	16	$91.6^{\pm 0.2}$
✓	✗	15	$96.7^{\pm 0.4}$
✓	✓	8	$\mathbf{100.0}^{\pm 0}$

Table 2: Effect of different SMoE gate functions on the heat diffusion dataset.

Gate	#Params	% within 1%
Fully-connected	50M	$85.2^{\pm 5.7}$
3×3 convolution	27	$91.3^{\pm 0.3}$
3×3 CoordConv [54]	81	$91.5^{\pm 0.4}$
3×3 LCN	110592	$96.3^{\pm 1.2}$
3×3 CoordConv$\times 3$	255	$91.6^{\pm 0.2}$
Tensor routing	12288	$\mathbf{100.0}^{\pm 0}$

This process is iterated to generate many timesteps from the given starting state. The dataset then consists of the generated timesteps for many different starting states.

If a model is able to learn the diffusion stencils and the location-diffusivity correspondence, it can exactly predict the next timestep. Further, the diffusivity stencils are also simple, and exactly correspond to a 3×3 convolution kernel with no nonlinearity.

The particular dataset we use consists of $100{,}000$ 64×64 samples, with $1{,}000$ initial states evolved for 100 timesteps each. Adding more samples did not significantly change results. There are three region types, with diffusivities 0.25, 0.025, and 0.0025. Figure 3 (center) shows the region map and diffusion stencils. We report results using "% within 1%", the percentage of locations in a sample that are within 1% relative error of the true value, as this is more interpretable than a mean-square error.

Results. Figure 3 (left) shows results on the heat diffusion dataset for SMoEs and a number of baselines: CNNs, LCNs, fully-connected (FC) layers, CoordConv [54], CondConv [88], LRLCNs [25], vision transformers (ViT) [21], and vision MoEs (V-MoE) [71]. For the LCN and FC layers, we use only a single layer as additional ones showed no benefit. Convolution, CoordConv, CondConv, and LRLCN are the best network found among a set with up to three layers, 3×3 kernels, 12 filters per layer, batchnorm, and ReLU activations. ViT and V-MoE use one transformer block with a patch size of 4×4, an embedding dimension of 128, and four heads. LRLCNs use three basis filters and an input-dependent combiner. We also tried unshared combining weights with no input dependence (LRLCN-ND). V-MoEs select one expert from a set of three. Our SMoEs use a single layer with three experts and select $E = 1$ expert per point. All models were trained with batch size 32, Adam [44] with a learning rate of 0.001 (decayed by $10\times$ after no validation improvement for 15 epochs), and early stopping after no improvement for 30 epochs. Additional hyperparameter tuning did not significantly improve results. We report SMoE parameters as expert+gate parameters.

SMoEs achieve perfect performance on this dataset. Further, by examining the learned routing and experts (Fig. 3 (right)), we can see that it has indeed correctly learned the diffusion stencils and location-dependence. LCNs also achieve this, but require $3\times$ more parameters, and require 110 epochs to converge (versus 8 for SMoEs). Fully-connected layers failed to learn the data well, likely due to the challenge of optimizing so many parameters. Other methods all converge to between 91 and 94 % within 1%. Examining their predictions and weights, we observe that they do not appear to have learned the location-dependence of the diffusivities, and instead converged to predicting with an "average" diffusivity across the domain. MoE methods (CondConv and V-MoE) also fail in this manner, as their coarse-grained experts are unable to specialize.

We now discuss a number of different ablations of the SMoE architecture and design.

What if the "right" expert configuration is not known? While in the above experiments, we were able to select the SMoE expert configuration (number of experts, number of selected experts, expert filter size) so that it is both necessary and sufficient to learn the task, in many situations this information may not be available. We considered alternative SMoE configurations varying each of these parameters: ❶ using six experts; ❷ experts with 5×5 kernels; ❸ and selecting two experts per location from six total. For case ❸, we summed the two SMoE output channels together.

In all three cases, the SMoE achieved $100.0^{\pm 0}$ % within 1% on the heat diffusion task. In ❶, we found that they learned duplicate diffusion stencils and still routed them appropriately. ❷ learned the five-point stencil plus a boundary of near-zero values, thus being nearly identical to the 3×3 kernel.

Table 3: WeatherBench [67] results (latitude-weighted RMSE).

Model	Z500 [m^2s^{-2}]		T850 [K]	
	3 days	5 days	3 days	5 days
Rasp and Thuerey [69]	$316^{\pm 2.4}$	$563^{\pm 3.1}$	$1.80^{\pm 0.02}$	$2.84^{\pm 0.03}$
SMoE after first layer	$305^{\pm 1.9}$	$556^{\pm 2.2}$	$1.77^{\pm 0.01}$	$2.80^{\pm 0.03}$
Last layer SMoE	$298^{\pm 2.6}$	$553^{\pm 3.2}$	$1.73^{\pm 0.02}$	$2.78^{\pm 0.04}$
3×3 convs→SMoE	$278^{\pm 2.0}$	$530^{\pm 1.8}$	$1.69^{\pm 0.01}$	$2.65^{\pm 0.01}$
+ gate prior	$\mathbf{270}^{\pm 1.9}$	$\mathbf{525}^{\pm 2.0}$	$\mathbf{1.66}^{\pm 0.02}$	$\mathbf{2.60}^{\pm 0.01}$

Table 4: ImageNet validation accuracy results.

Model	Top-1 %	Top-5 %
ResNet-50 [34, 82]	$80.83^{\pm 0.04}$	$95.39^{\pm 0.03}$
LRLCN [25]	$80.90^{\pm 0.02}$	$95.41^{\pm 0.05}$
SMoE after first layer	$80.85^{\pm 0.05}$	$95.40^{\pm 0.01}$
Last layer SMoE	$80.91^{\pm 0.04}$	$95.42^{\pm 0.03}$
3×3 convs→SMoE	$81.33^{\pm 0.03}$	$95.52^{\pm 0.01}$
Wide ResNet-50-2 [89]	$\mathbf{81.76}^{\pm 0.03}$	$\mathbf{95.74}^{\pm 0.02}$

Finally, ❸ learned diffusion stencils that summed together to produce the correct diffusivity. Thus, we can see that SMoEs are robust and adapt well to these sorts of architecture choices.

RC loss and expert error damping. Table 1 shows results for training SMoEs with and without our routing classification loss (§2.2) and expert error damping (§2.3). Without the RC loss, SMoE performance is at par with other baselines in Fig. 3, but once it is added, performance improves significantly as the gating function now learns the location-dependency in the data. Adding expert error damping further improves performance and convergence by limiting the impact of gate misroutings on expert learning. However, damping on its own offers little benefit, as it does not improve gate learning. These results show that these refinements are critical for good performance.

Gate function. Table 2 shows the performance of different gating functions (§2.1) on the SMoE. We consider six options: A single fully-connected layer (as is commonly used in MoEs [71, 75]); a single 3×3 convolution, CoordConv [54], or LCN layer; a gate with three CoordConv layers with batchnorm and ReLU; and our tensor routing gate. When training, we also considered auxiliary losses and other methods for improving performance (see Appendix C) and report the best result. Our tensor routing offers the best performance. An LCN performs second-best, likely because it also uses separate parameters per location, but uses $9\times$ as many parameters and requires significant computation. Other methods do not appear able to effectively capture location-dependence.

Other ablations. We conduct a number of additional ablation studies in Appendix C, including using auxiliary losses and routing noise during training, routing normalizations, and expert functions.

3.2 Medium-Range Weather Prediction

We now discuss results on the WeatherBench [67] medium-range weather forecasting benchmark. This benchmark uses the ERA5 reanalysis dataset [35], with hourly global weather data for 1979–2018. We use the data subset suggested by Rasp et al. [67] at 5.625° resolution (32×64 grid points) and train on data from 1979–2015, validate on 2016, and report test results for 2017–2018; we do not use additional pretraining data. We otherwise follow the training methodology of Rasp and Thuerey [69]. The target quantities to predict are geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) with a three- and five-day lead time. We report results using latitude-weighted root-mean-square error (RMSE).

As a baseline, we use the ResNet architecture [34] introduced by Rasp and Thuerey [69], which currently reports the best results on WeatherBench. This architecture consists of 19 residual blocks each with two [3×3 convolution → LeakyReLU → batchnorm → dropout] layers, plus an initial 7×7 convolution layer. All convolutions but the last have 128 filters.

We adapt this network to use SMoEs by considering three different configurations: adding an SMoE layer after the first convolution; adding an SMoE layer after the final convolutional layer; and replacing all 3×3 convolutions with SMoE layers. Each SMoE has selects the same number of experts as the original layer had filters, and has twice as many experts (i.e., $|\mathcal{E}| = 256$ and $E = 128$). We also *share* the tensor routing gate across all SMoE layers with the same number of experts, so the overhead of the gate is minimal.

Because the weather data is on a fixed grid with underlying location-dependence (the Earth), we expect SMoEs to convey some benefit by specializing to the characteristics of different regions. In Table 3, we observe that this is indeed the case, and adding SMoEs improves results in all situations, with the most significant improvement coming through replacing all 3×3 convolution layers with SMoEs. This showcases the advantage of incorporating appropriate location-dependent biases.

Incorporating prior knowledge into gates. While the exact nature of the location-dependence of this data is unknown, we do have a broad prior on some aspects of it, such as whether a point is land or sea. This information can be incorporated into an SMoE by initializing the tensor routing gate using it to bias routing to different experts. To this end, we use the land-sea mask from ERA5 to initialze the gate to route land locations to half the experts and sea locations to the other half. Note this does not fix the routing, as the gate is able to adjust as it learns. Further, the land-sea mask is already included in the input data, so all models already had access to this information.

Results with this are in the "+ gate prior" line of Table 3, and perform best. This configuration sets a new state-of-the-art for WeatherBench when not using additional data. Indeed, it nearly matches the performance of a ResNet with 150 years of additional pretraining data [69].

3.3 Image Classification

Lastly, we present results on several image classification tasks; we focus here on ImageNet-1k [19] and discuss results on additional datasets in Appendix D. While these datasets do not have a strict location-dependent structure, relaxing the strict translation equivariance of convolutions can bring benefits, and enables a direct comparison with Elsayed et al. [25]. We follow their experimental methodology and train using the recipe of Vryniotis [82]. We either insert an SMoE layer after the first or last convolutional layer of ResNet-50 [34] or replace all 3×3 convolutions with SMoE layers. Our SMoEs have twice as many experts as the original convolution layer and select half of them, to keep output dimensions constant. Gating layers are shared among all equally-sized blocks. For comparison, we also train ResNet-50 with all 3×3 convolutions replaced by LRLCN [25] layers; and a Wide ResNet-50-2 [89], which has comparable parameters to SMoEs.

Table 4 shows that SMoEs outperform LRLCNs when we replace all 3×3 convolutions, while using 56% of the parameters. However, a wide ResNet performs best overall. Nevertheless, this shows that ImageNet classification does indeed benefit from relaxing translation equivariance.

4 Discussion

We presented the Spatial Mixture-of-Experts layer, a novel layer that learns underlying location dependencies in data and then uses fine-grained routing to specialize experts to different areas. We also introduce a routing classification loss and expert error damping, which enable SMoEs to perform well on dense regression tasks. Prior work shows limited effectiveness on these tasks: Either it does not capture location-dependence (e.g., convolutions) or it operates at a coarse-grained level (e.g., standard MoEs). By overcoming these challenges, we demonstrate a new capability for neural networks, and set a new state-of-the-art for medium-range weather prediction.

Many other problems of broad societal import have a similar spatial structure, particularly in scientific domains [8], and we expect SMoEs to be applicable to them. However, tasks such as facial recognition and surveillance have also historically shown benefit from such improvements [80] and SMoEs should therefore be used with care.

SMoEs show that learning location-dependence is a powerful inductive bias for certain types of data, and there are many avenues for further study. Two key areas of particular interest are to develop improved implementations for fine-grained, sparse routing; and to generalize SMoEs from operating on grids to general graphs, which would enable them to be applied to many additional tasks.

References

[1] Alhabib Abbas and Yiannis Andreopoulos. 2020. Biased mixtures of experts: Enabling computer vision inference under data transfer limitations. *IEEE Transactions on Image Processing* 29 (2020). arXiv:2008.09662 [cs.LG]

[2] Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. 2019. Machine learning for precipitation nowcasting from radar images. (2019). arXiv:1912.12132 [cs.CV]

[3] Subutai Ahmad and Luiz Scheinkman. 2019. How can we be so dense? The benefits of using highly sparse representations. (2019). arXiv:1903.11257 [cs.LG]

[4] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. 2016. Network of experts for large-scale image categorization. In *European Conference on Computer Vision (ECCV)*. arXiv:1604.06119 [cs.CV]

[5] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. 2016. Dynamic capacity networks. In *International Conference on Machine Learning (ICML)*. arXiv:1511.07838 [cs.LG]

[6] American Meteorological Society. 2022. Medium-range forecast. https://glossary.ametsoc.org/wiki/Medium-range_forecast

[7] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient Large Scale Language Modeling with Mixtures of Experts. arXiv:2112.10684 [cs.CL]

[8] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. 2019. *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*. Technical Report. United States Department of Energy, Office of Science.

[9] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2016. Conditional computation in neural networks for faster models. In *International Conference on Learning Representations Workshops (ICLR-W)*. arXiv:1511.06297 [cs.LG]

[10] Yoshua Bengio. 2013. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*. arXiv:1305.0445 [cs.LG]

[11] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. (2013). arXiv:1308.3432 [cs.LG]

[12] Ke Chen, Lei Xu, and Huisheng Chi. 1999. Improved learning algorithms for mixture of experts in multiclass classification. *Neural networks* 12, 9 (1999).

[13] Yu-hsin Chen, Ignacio Lopez Moreno, Tara Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada. 2015. Locally-connected and convolutional neural networks for small footprint speaker recognition. In *INTERSPEECH*.

[14] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. 2019. You look twice: GaterNet for dynamic filter selection in CNNs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:1811.11205 [cs.CV]

[15] Kyunghyun Cho and Yoshua Bengio. 2014. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. (2014). arXiv:1406.7362 [stat.ML]

[16] Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. 2022. Unified Scaling Laws for Routed Language Models. arXiv:2202.01169 [cs.CL]

[17] Shail Dave, Riyadh Baghdadi, Tony Nowatzki, Sasikanth Avancha, Aviral Shrivastava, and Baoxin Li. 2021. Hardware acceleration of sparse and irregular tensor computations of ML models: A survey and insights. *Proc. IEEE* 109, 10 (2021). arXiv:2007.00864 [cs.AR]

[18] Andrew Davis and Itamar Arel. 2013. Low-rank approximations for conditional feedforward computation in deep neural networks. In *International Conference on Learning Representations Workshops (ICLR-W)*. arXiv:1312.4461 [cs.LG]

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Conference on computer vision and pattern recognition (CVPR)*.

[20] Ludovic Denoyer and Patrick Gallinari. 2014. Deep sequential neural network. (2014). arXiv:1410.0510 [cs.LG]

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*. arXiv:2010.11929 [cs.CV]

[22] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2021. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. arXiv:2112.06905 [cs.CL]

[23] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. 2021. ConViT: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning (ICML)*. arXiv:2103.10697 [cs.CV]

[24] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. (2013). arXiv:1312.4314 [cs.LG]

[25] Gamaleldin Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. 2020. Revisiting spatial invariance with low-rank local connectivity. In *International Conference on Machine Learning (ICML)*. arXiv:2002.02959 [cs.CV]

[26] Lasse Espeholt, Shreya Agrawal, Casper Sønderby, Manoj Kumar, Jonathan Heek, Carla Bromberg, Cenk Gazen, Jason Hickey, Aaron Bell, and Nal Kalchbrenner. 2021. Skillful Twelve Hour Precipitation Forecasts using Large Context Neural Networks. (2021). arXiv:2111.07470 [cs.LG]

[27] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv:2101.03961 [cs.LG]

[28] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. PathNet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* (2017). arXiv:1701.08734 [cs.NE]

[29] Hiroshi Fukui, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. 2019. Attention branch network: Learning of attention mechanism for visual explanation. In *Conference on computer vision and pattern recognition (CVPR)*. arXiv:1812.10025 [cs.CV]

[30] Karo Gregor and Yann LeCun. 2010. Emergence of complex-like cells in a temporal product network with local receptive fields. *arXiv preprint arXiv:1006.0448* (2010). arXiv:1006.0448 [cs.NE]

[31] Peter Grönquist, Chengyuan Yao, Tal Ben-Nun, Nikoli Dryden, Peter Dueben, Shigang Li, and Torsten Hoefler. 2021. Deep learning for post-processing ensemble weather forecasts. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021). arXiv:2005.08748 [cs.LG]

[32] Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. 2017. Hard mixtures of experts for large scale weakly supervised vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:1704.06363 [cs.CV]

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *International conference on computer vision (ICCV)*. arXiv:1502.01852 [cs.CV]

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition (CVPR)*. arXiv:1512.03385 [cs.CV]

[35] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. 2020. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society* 146, 730 (2020).

[36] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research* 22, 241 (2021). arXiv:2102.00554 [cs.LG]

[37] Gary B Huang, Honglak Lee, and Erik Learned-Miller. 2012. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Kevin Lee Hunter, Lawrence Spracklen, and Subutai Ahmad. 2021. Two Sparsities Are Better Than One: Unlocking the Performance Benefits of Sparse-Sparse Networks. (2021). arXiv:2112.13896 [cs.LG]

[39] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. 2021. Data movement is all you need: A case study on optimizing transformers. In *Conference on Machine Learning and Systems (MLSys)*. arXiv:2007.00072 [cs.LG]

[40] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991).

[41] Saumya Jetley, Nicholas A Lord, Namhoon Lee, and Philip HS Torr. 2018. Learn to pay attention. In *International Conference on Learning Representations (ICLR)*. arXiv:1804.02391 [cs.CV]

[42] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. [n.d.]. Dynamic filter networks. *Advances in neural information processing systems (NeurIPS)* ([n. d.]). arXiv:1605.09673 [cs.LG]

[43] Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural computation* 6, 2 (1994).

[44] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. arXiv:1412.6980 [cs.LG]

[45] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. (2009).

[46] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. 2020. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning (ICML)*.

[47] Jeffrey K Lazo, Rebecca E Morss, and Julie L Demuth. 2009. 300 billion served: Sources, perceptions, uses, and values of weather forecasts. *Bulletin of the American Meteorological Society* 90, 6 (2009).

[48] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998).

[49] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*. arXiv:2006.16668 [cs.CL]

[50] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. BASE layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning (ICML)*. arXiv:2103.16716 [cs.CL]

[51] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. SwinIR: Image restoration using swin transformer. In *International Conference on Computer Vision (ICCV)*. arXiv:2108.10257 [eess.IV]

[52] Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. 2020. Dynamic model pruning with feedback. In *International Conference on Learning Representations (ICLR)*. arXiv:2006.07253 [cs.LG]

[53] Drew Linsley, Dan Shiebler, Sven Eberhardt, and Thomas Serre. 2019. Learning what and where to attend. In *International Conference on Learning Representations (ICLR)*. arXiv:1805.08819 [cs.CV]

[54] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An intriguing failing of convolutional neural networks and the CoordConv solution. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv:1807.03247 [cs.CV]

[55] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision (ICCV)*. arXiv:2103.14030 [cs.CV]

[56] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *International conference on computer vision (ICCV)*. arXiv:1411.7766 [cs.CV]

[57] E Majani, Ruth Erlanson, and Yaser Abu-Mostafa. 1988. On the K-winners-take-all network. In *Advances in neural information processing systems (NeurIPS)*.

[58] Alireza Makhzani and Brendan J Frey. 2015. Winner-take-all autoencoders. In *Advances in neural information processing systems (NeurIPS)*. arXiv:1409.2752 [cs.LG]

[59] Mason McGill and Pietro Perona. 2017. Deciding how to decide: Dynamic routing in artificial neural networks. In *International Conference on Machine Learning (ICML)*. arXiv:1703.06217 [stat.ML]

[60] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang Koh, Quoc Le, and Andrew Ng. 2010. Tiled convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)* (2010).

[61] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2019. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations (ICLR)*. arXiv:1810.05148 [stat.ML]

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*. arXiv:1912.01703 [cs.LG]

[63] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. [n.d.]. FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators. ([n. d.]). arXiv:2202.11214

[64] Svetlana Pavlitskaya, Christian Hubschneider, Michael Weber, Ruby Moritz, Fabian Huger, Peter Schlicht, and Marius Zollner. 2020. Using mixture of expert models to gain insights into semantic segmentation. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[65] Prajit Ramachandran and Quoc V Le. 2018. Diversity and depth in per-example routing models. In *International Conference on Learning Representations (ICLR)*.

[66] Marc'Aurelio Ranzato, Joshua Susskind, Volodymyr Mnih, and Geoffrey Hinton. 2011. On deep generative models with applications to recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[67] Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. 2020. WeatherBench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems* 12, 11 (2020). arXiv:2002.00469 [physics.ao-ph]

[68] Stephan Rasp and Sebastian Lerch. 2018. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review* 146, 11 (2018). arXiv:1805.09091 [stat.ML]

[69] Stephan Rasp and Nils Thuerey. 2021. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for WeatherBench. *Journal of Advances in Modeling Earth Systems* 13, 2 (2021). arXiv:2008.08626 [physics.ao-ph]

[70] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. 2021. Skilful precipitation nowcasting using deep generative models of radar. *Nature* 597, 7878 (2021). arXiv:2104.00954 [cs.LG]

[71] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling Vision with Sparse Mixture of Experts. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv:2106.05974 [cs.CV]

[72] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2018. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *International Conference on Learning Representations (ICLR)*. arXiv:1711.01239 [cs.LG]

[73] Christoph Schär, Oliver Fuhrer, Andrea Arteaga, Nikolina Ban, Christophe Charpilloz, Salvatore Di Girolamo, Laureline Hentgen, Torsten Hoefler, Xavier Lapillonne, David Leutwyler, et al. 2020. Kilometer-scale climate models: Prospects and challenges. *Bulletin of the American Meteorological Society* 101, 5 (2020).

[74] MG Schultz, Clara Betancourt, Bing Gong, Felix Kleinert, Michael Langguth, LH Leufen, Amirpasha Mozaffari, and Scarlet Stadtler. 2021. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A* 379, 2194 (2021).

[75] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*. arXiv:1701.06538 [cs.LG]

[76] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems (NeurIPS)*. arXiv:1506.04214 [cs.CV]

[77] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. 2017. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Advances in neural information processing systems (NeurIPS)*. arXiv:1706.03458 [cs.CV]

[78] Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. 2020. MetNet: A neural weather model for precipitation forecasting. (2020). arXiv:2003.12140 [cs.LG]

[79] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. 2017. Training sparse neural networks. In *Conference on computer vision and pattern recognition workshops (CVPRW)*. arXiv:1611.06694 [cs.CV]

[80] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[81] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. 2021. Scaling local self-attention for parameter efficient visual backbones. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:2103.12731 [cs.CV]

[82] year=2021 url=https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/ Vryniotis, Vasilis. [n.d.]. How to Train State-of-the-Art Models Using TorchVision's Latest Primitives.

[83] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. 2017. Residual attention network for image classification. In *Conference on computer vision and pattern recognition (CVPR)*. arXiv:1704.06904 [cs.CV]

[84] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *International Conference on Computer Vision (ICCV)*. arXiv:2102.12122 [cs.CV]

[85] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2022. PVT v2: Improved baselines with Pyramid Vision Transformer. *Computational Visual Media* (2022). arXiv:2106.13797 [cs.CV]

[86] Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. 2020. Deep mixture of experts via shallow embedding. In *Uncertainty in Artificial Intelligence*. arXiv:1806.01531 [cs.CV]

[87] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. CBAM: Convolutional block attention module. In *European conference on computer vision (ECCV)*. arXiv:1807.06521 [cs.CV]

[88] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. 2019. CondConv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv:1904.04971 [cs.CV]

[89] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *British Machine Vision Conference (BMVC)*. arXiv:1605.07146 [cs.CV]

[90] Kaili Zhao, Wen-Sheng Chu, and Honggang Zhang. 2016. Deep region and multi-label learning for facial action unit detection. In *Conference on computer vision and pattern recognition (CVPR)*.

[91] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. arXiv:2202.08906 [cs.CL]

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the key assumption that all data must have similar location-dependence for SMOEs to be applicable in §1. We discuss other assumptions in §2 and implementation details in §2.4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss this in §4.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include code and instructions in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] §3 discusses general training details and Appendix A provides additional details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results in §3 were run multiple times, and we report the mean and standard deviation.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This is discussed in §3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Most datasets we use are publicly available. Our heat diffusion dataset is synthetic and can be generated by our included code. The supplementary material includes the license for our code.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include code to implement SMOEs and run experiments.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] Some image classification datasets we present results on, notably ImageNet, contain images of people. We have not obtained consent from them. However, these are standard, widely-used benchmark datasets for image classification, and we do not include any results that disclose personal information. We do not introduce or curate any new data that contains personal information.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We do not introduce or curate any new data that contains personal information or offensive content. We do use ImageNet as a standard benchmark, which may contain personally identifiable information (pictures of people), but do not report any results that include personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]