



Università
Ca' Foscari
Venezia

Master's Degree Programme

in Computer Science and Information Technology
Artificial Intelligence and Data Engineering

Master's Thesis

Attention as a Kernel:

**Uncovering Latent Geometry
for Zero-Shot Transfer**

Supervisor

Chiar.mo prof. Andrea Torsello

Candidate

Soroush Karami

Matricola 904132

Academic Year

2025/2026

Abstract

Attention is usually treated as an internal computation of a neural network: the matrix of pairwise weights it produces is used to mix information and then discarded. In this thesis we argue that the similarity structure attention computes over the data can instead be turned into a kernel. We compute attention between samples on top of a frozen convolutional backbone, so that any success is attributable to attention alone, and build a kernel from the output of this inter-sample attention layer, a Gram matrix that is a valid kernel by construction, which we feed to a Support Vector Machine. With a single, unified method we evaluate this attention-induced kernel in three settings: multi-task, transfer (which plays the role of zero-shot), and few-shot, on the genuinely multi-attribute datasets CelebA and AwA2. We find that the kernel matches, and sometimes beats, the network it is derived from, and that the geometry it captures carries over to tasks the encoder never saw, evidence that attention encodes a transferable latent structure over the data.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Andrea Torsello, who accompanied me step by step throughout this project. He generously gave me his time whenever I needed it, in any circumstance, even during his holidays, and his guidance was always clear and insightful. Beyond his help with the work itself, he taught me a great deal along the way. Perhaps the most valuable lesson was how to think: he showed me how to look at a research problem from the right perspective and how to work through the obstacles in its path. For his patience, his generosity, and everything he taught me, I am truly grateful.

Contents

Abstract	i
Acknowledgements	ii
1 Introduction	1
2 Background and Related Work	4
2.1 Neural Networks and Feature Extraction	4
2.2 The Attention Mechanism	6
2.2.1 Scaled dot-product attention	6
2.2.2 Sample-level attention with tied queries and keys	7
2.3 Kernel Methods	8
2.4 Support Vector Machines	9
2.4.1 The maximum-margin hyperplane	9
2.4.2 Soft margin	9
2.4.3 The dual problem and the kernel	9
2.5 Related Work	11
2.5.1 Attention and what it represents	11
2.5.2 Kernels and neural networks	11
2.5.3 Zero-shot and transfer learning	12
2.6 Summary	12
3 Proposed Method	14
3.1 Overview	14
3.2 Design rationale: the path to the final method	15
3.2.1 Kernel from embedding inner products	15
3.2.2 Novelty-based zero-shot	15
3.2.3 Semantic projection for zero-shot	16
3.2.4 Resolution: transfer as zero-shot	16
3.3 Stage 1: Frozen backbone	16

3.4	Stage 2: Full inter-sample attention	17
3.5	Stage 3: Kernel construction	18
3.6	Stage 4: Classification with an SVM	18
3.7	The complete algorithm	19
3.8	Transfer as zero-shot, and a structural limit	19
4	Experiments and Results	21
4.1	Datasets and Tasks	21
4.2	Evaluation Protocol	22
4.3	Implementation details	23
4.4	Results	24
4.5	The limit of label-free zero-shot	26
4.6	Limitations	28
4.7	Summary	28
5	Conclusion and Future Work	29
5.1	Summary of contributions	29
5.2	Future work	30

Chapter 1

Introduction

Over the past decade, computer vision has been reshaped by deep neural networks that learn their own representations of the visual world directly from data. A representation learned this way is useful not only for the task it was trained on: the features it produces often carry information that helps on other problems too. This reusability, which we call *transfer*, is one of the reasons deep representations have become the foundation of modern vision systems. In a growing number of these systems, the part that relates and combines information is the *attention* mechanism [1].

Attention works by computing, for each element of its input, a weighted combination of the others, where the weights say how relevant each pair of elements is. Those pairwise weights form a matrix, and that matrix is really a similarity structure: it records, for every pair, how strongly the model treats them as related. When attention is applied across a collection of samples rather than within a single input, the matrix describes a geometry over the dataset itself, one in which samples the model finds similar are pulled together and dissimilar ones are pushed apart. Normally this matrix stays hidden inside the network. It is produced, used to mix information, and then thrown away, and the geometry it encodes is never looked at on its own.

Kernel methods are built on exactly this kind of object. A kernel is a matrix of pairwise similarities that implicitly defines a geometry over the data, and classifiers such as the Support Vector Machine (SVM) [2] work directly on top of that geometry, without ever needing an explicit feature representation. Put next to each other, an attention matrix and a kernel matrix have a lot in common: both are square, and both express a notion of similarity between samples. That resemblance is where this thesis starts. If attention already induces a similarity structure over samples, can that structure be turned into a kernel and used to

drive a kernel classifier?

Our answer is yes. We build the kernel of an SVM from an attention layer that operates between samples. To be sure that any success we see comes from attention and not from the strength of a pretrained feature extractor, we keep a convolutional backbone frozen throughout: each image is mapped to a fixed feature vector by a ResNet [3] that receives no further training, and the only component that is actually learned is a sample-level attention layer on top of those features. The kernel we classify with is a Gram matrix built from the output of that attention layer, so the attention computation sits at the heart of the similarity we use rather than at its edge. We call this construction *attention as a kernel*.

Our experiments show that the kernel attention induces is much more than a by-product of training. On datasets whose samples carry several independent labels, that is, genuinely multi-task datasets, the attention kernel fed to an SVM matches, and sometimes beats, the accuracy of the very network it was taken from. That is already worth pausing on: a similarity matrix, read out from inside a model and handed to a classical classifier, holds its own against the model itself. The more important result, though, is about *latent geometry*. The structure attention captures is not tied to the task it was trained on. If we train the attention encoder without ever showing it a target task and then apply its kernel to that unseen task, classification still works; because the encoder never saw the target task, this setting acts as a form of *zero-shot* transfer. The same kernel also stays useful when only a handful of labelled examples per class are available, a *few-shot* setting. So a single, unified method, an attention-induced kernel followed by an SVM, covers three different evaluation settings without any change.

The main contributions of this thesis are the following:

- We formulate *attention as a kernel*: a construction that turns the output of a sample-level attention layer into a valid kernel for an SVM. Freezing the backbone isolates the contribution of the attention layer, so that the performance we measure reflects the geometry attention induces rather than the representational power of the feature extractor.
- We evaluate this one method across three settings, multi-task, transfer (which we argue plays the role of zero-shot), and few-shot classification, on two datasets, CelebA [4] and AwA2 [5], whose samples are annotated with several genuinely independent attributes.

- We show that the attention-induced kernel matches or surpasses the trained network on its own tasks, and, more importantly, that the geometry it encodes carries over to tasks the encoder never saw, evidence that attention captures a transferable latent structure over the data.

The rest of the thesis is organised as follows. Chapter 2 covers the background needed to follow the method, reviewing neural networks, the attention mechanism, kernels, and the Support Vector Machine from first principles, and places the approach among related work on attention, neural kernels, and zero-shot transfer. Chapter 3 presents the method itself, describing the four-stage pipeline (frozen feature extraction, sample-level attention, kernel construction, and SVM classification) and illustrating the overall architecture. Chapter 4 reports the experiments across the three settings, along with the datasets, the training procedure, and a comparison against the network baseline. Chapter 5 concludes and outlines directions for future work.

Chapter 2

Background and Related Work

This chapter gathers the tools the method rests on. The whole approach is really a short sentence in four words: frozen features, attention, kernel, classifier. Each of them matches one section below. Section 2.1 reviews neural networks, and in particular the use of a pretrained network as a fixed feature extractor, which is what lets us isolate the contribution of attention. Section 2.2 introduces the attention mechanism, first in the standard Transformer form and then in the sample-level form we adopt, which is the object we later turn into a kernel. Section 2.3 defines kernels and explains why a matrix of pairwise similarities can act as one. Section 2.4 describes the Support Vector Machine, the classifier that consumes our kernel, and shows that it touches the data only through the kernel matrix, which is the property our method relies on. Section 2.5 places the work among related research.

2.1 Neural Networks and Feature Extraction

A feed-forward neural network is a parametric function built by composing simple layers. A single fully connected layer maps an input vector $\mathbf{x} \in \mathbb{R}^d$ to

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (2.1)$$

where $\mathbf{W}$ and $\mathbf{b}$ are a learnable weight matrix and bias vector, and $\sigma(\cdot)$ is a non-linear activation applied element-wise, such as the rectified linear unit $\text{ReLU}(z) = \max(0, z)$. Stacking L such layers gives a network $f_\theta = f^{(L)} \circ \dots \circ f^{(1)}$ with parameters $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^L$. Composing non-linear layers is what lets the network approximate complicated mappings from inputs to outputs [6].

The parameters are fit by minimising an empirical risk. Given a labelled

dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and a loss ℓ , training solves

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{x}_i), y_i), \quad (2.2)$$

usually by stochastic gradient descent: the gradient of the loss with respect to θ is computed by back-propagation, and the parameters are moved along its negative direction.

Convolutional networks. For images, fully connected layers ignore spatial structure and scale badly with resolution. Convolutional neural networks (CNNs) replace the dense multiplication in (2.1) with a convolution by a small, shared kernel, so the same local filter is applied across the whole image. A convolutional layer produces feature maps by sliding learnable filters over its input, and interleaving convolutions, non-linearities and spatial down-sampling builds a hierarchy of representations: edges and textures in the early layers, object parts and semantic concepts in the deep ones [7].

Residual networks. Simply adding depth tends to hurt accuracy, because gradients become hard to push through many layers. Residual networks (ResNets) fix this with shortcut connections that let a block learn a residual relative to its input:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{\mathbf{W}_i\}) + \mathbf{x}, \quad (2.3)$$

where $\mathcal{F}$ is a few stacked convolutional layers. If the best mapping is close to the identity, driving $\mathcal{F}$ towards zero is easier than fitting an identity with a stack of non-linear layers. This one change makes it possible to train networks tens or hundreds of layers deep, and it has made the ResNet family a standard backbone in computer vision [3].

Pretrained backbones and feature extraction. A network trained on a large dataset such as ImageNet learns representations that are useful well beyond its original task, because the deep features it produces tend to capture generic visual structure. This is the idea behind transfer learning. In its simplest form, one takes a pretrained network, removes its classification head, and uses the remaining layers as a fixed map

$$\phi: \mathcal{X} \longrightarrow \mathbb{R}^d, \quad \mathbf{x} \mapsto \phi(\mathbf{x}), \quad (2.4)$$

that turns each image into a feature vector. When the backbone is *frozen*, its weights get no gradient updates, so ϕ is identical before and after any downstream training. This has a consequence that matters throughout the thesis: with the feature extractor held fixed, any change in performance from a component placed on top of ϕ can be credited to that component, not to the backbone adapting to the task. We use exactly this to keep our results attributable to the attention layer rather than to the representational power of the backbone.

2.2 The Attention Mechanism

Attention relates the elements of a collection by letting each element gather information from the others in proportion to how relevant they are [1]. We first give the standard form and then the specific form we use.

2.2.1 Scaled dot-product attention

The standard formulation works on three matrices: a set of *queries* $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, a set of *keys* $\mathbf{K} \in \mathbb{R}^{m \times d_k}$, and a set of *values* $\mathbf{V} \in \mathbb{R}^{m \times d_v}$, where each row is the query, key or value of one element. The output is

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V}, \quad (2.5)$$

with the softmax applied independently to each row. Reading (2.5) from the inside out makes it concrete. The product $\mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{n \times m}$ holds, in entry (i, j) , the dot product between query i and key j , a raw measure of how compatible the two elements are. Dividing by $\sqrt{d_k}$ stops these scores from growing with the dimension and keeps the softmax from saturating. The row-wise softmax then turns each row of scores into a probability distribution, giving the *attention matrix*

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{n \times m}, \quad A_{ij} \geq 0, \quad \sum_j A_{ij} = 1. \quad (2.6)$$

Entry A_{ij} is the weight element i places on element j , and $\mathbf{A}\mathbf{V}$ replaces each element by a convex combination of the values, weighted by attention.

In *self-attention*, the queries, keys and values are all linear projections of a single input matrix $\mathbf{X}$:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \quad (2.7)$$

with learnable projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$. In practice several attention functions are run in parallel and concatenated, which is called multi-head attention and lets the model attend to different kinds of relationships at once; for our purposes the single-head form in (2.5) is enough.

The point this thesis builds on is already visible in (2.6): before anything else, the attention matrix $\mathbf{A}$ is a matrix of pairwise scores between elements. It records how strongly the model relates every pair, which is exactly the information a similarity measure carries. Usually $\mathbf{A}$ is just an intermediate quantity, used to form $\mathbf{AV}$ and then discarded; we instead take the similarity structure it expresses seriously in its own right.

2.2.2 Sample-level attention with tied queries and keys

Two departures from the standard setting matter for our method. First, we apply attention *across samples* rather than across the patches or tokens of a single input: the rows of $\mathbf{X}$ are the feature vectors $\phi(\mathbf{x}_i)$ of different images, produced by the frozen backbone of (2.4), so the attention matrix describes relationships within a collection of samples. Second, we *tie* the query and key projections, $\mathbf{W}_Q = \mathbf{W}_K$, so that $\mathbf{Q} = \mathbf{K}$. Writing $\mathbf{K} = \mathbf{XW}_K$ and $\mathbf{V} = \mathbf{XW}_V$, the attention becomes

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{KK}^\top}{\sqrt{d}}\right), \quad \mathbf{Z} = \mathbf{AV}, \quad (2.8)$$

where d is the head dimension. Tying queries and keys makes the pre-softmax score matrix $\mathbf{KK}^\top$ symmetric, so the raw compatibility between two samples does not depend on which one is treated as the query, and it also halves the number of projection parameters. The row-wise softmax in (2.8) does not preserve this symmetry, since each row is normalised by its own sum, so the attention weight matrix $\mathbf{A}$ is in general not symmetric. That is fine for what follows, because the object we eventually hand to the classifier is not $\mathbf{A}$: as Section 2.3 shows, our kernel is a Gram matrix built from the attention output, and it is symmetric and positive semidefinite by construction whether or not $\mathbf{A}$ is symmetric. The value projection $\mathbf{V}$ plays its usual role, carrying the content that attention mixes, and it takes part in learning. The output $\mathbf{Z}$ of (2.8) is the quantity our kernel is built from; Chapter 3 develops the full construction.

2.3 Kernel Methods

Kernel methods apply linear algorithms in a possibly very high-dimensional feature space without ever computing coordinates in that space [8, 9]. Suppose a feature map

$$\varphi: \mathcal{X} \longrightarrow \mathcal{H} \quad (2.9)$$

sends inputs into an inner-product space $\mathcal{H}$. A *kernel* is the function that returns inner products in that space,

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}. \quad (2.10)$$

Their value comes from the *kernel trick*: any algorithm that touches the data only through inner products $\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$ can be run entirely in terms of k , so φ never has to be evaluated explicitly. This makes implicit, even infinite-dimensional, feature spaces tractable.

Given a finite sample $\{\mathbf{x}_i\}_{i=1}^N$, a kernel is represented by the *Gram matrix* (or kernel matrix)

$$\mathbf{K} \in \mathbb{R}^{N \times N}, \quad K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \quad (2.11)$$

For k to be a valid kernel, that is, to correspond to an inner product in some space via (2.10), the Gram matrix must be *symmetric* and *positive semidefinite* (PSD): for every finite sample and every vector $\mathbf{c} \in \mathbb{R}^N$,

$$K_{ij} = K_{ji} \quad \text{and} \quad \mathbf{c}^\top \mathbf{K} \mathbf{c} = \sum_{i,j} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \quad (2.12)$$

The condition is also sufficient: by Mercer's theorem, any symmetric PSD kernel can be written as an inner product of feature maps in a suitable reproducing-kernel Hilbert space [8]. Familiar examples are the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$, the polynomial kernel $(\mathbf{x}^\top \mathbf{x}' + c)^p$, and the Gaussian (RBF) kernel $\exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$.

Two things follow from (2.12) that we lean on later. First, a kernel method never needs an explicit feature representation: it is enough to supply a valid $N \times N$ matrix of pairwise similarities. Any symmetric PSD matrix can therefore be handed to the classifier as a *precomputed* kernel, no matter how it was obtained, in our case from attention rather than from a closed-form kernel function. Second, symmetry and positive semidefiniteness are easy to get by construction. If we

form a matrix as a Gram matrix of some rows $\mathbf{Z} \in \mathbb{R}^{N \times d}$,

$$\mathbf{K} = \mathbf{Z}\mathbf{Z}^\top, \quad \mathbf{c}^\top \mathbf{K} \mathbf{c} = \|\mathbf{Z}^\top \mathbf{c}\|^2 \geq 0, \quad (2.13)$$

the result is symmetric and PSD automatically. This is the construction we apply to the attention output $\mathbf{Z}$ of (2.8), and it is what makes the attention-induced matrix a legitimate kernel.

2.4 Support Vector Machines

The Support Vector Machine (SVM) is the classifier we pair with our kernel. It is a natural choice because, as we show below, it touches the data only through the kernel matrix [2, 10].

2.4.1 The maximum-margin hyperplane

Consider binary classification with labels $y_i \in \{-1, +1\}$. A linear classifier separates the classes with a hyperplane $\mathbf{w}^\top \mathbf{x} + b = 0$ and predicts the sign of $\mathbf{w}^\top \mathbf{x} + b$. When the data are linearly separable many such hyperplanes exist, and the SVM picks the one that maximises the *margin*, the distance to the nearest points of either class. Since the margin equals $2/\|\mathbf{w}\|$, maximising it is equivalent to

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, N. \quad (2.14)$$

2.4.2 Soft margin

Real data are rarely perfectly separable, so slack variables $\xi_i \geq 0$ allow margin violations, penalised by a constant $C > 0$ that trades margin width against training errors:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad \text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \quad (2.15)$$

2.4.3 The dual problem and the kernel

Introducing Lagrange multipliers $\alpha_i \geq 0$ for the margin constraints and $\mu_i \geq 0$ for the non-negativity of the slacks gives the Lagrangian

$$\mathcal{L} = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i] - \sum_i \mu_i \xi_i. \quad (2.16)$$

Setting the derivatives with respect to $\mathbf{w}$, b and ξ_i to zero gives the stationarity conditions

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i, \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i + \mu_i = C. \quad (2.17)$$

Substituting these back removes the primal variables and, together with $\mu_i \geq 0$, gives the *dual problem*

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C. \quad (2.18)$$

The solution satisfies the Karush–Kuhn–Tucker (KKT) conditions; in particular the complementary-slackness conditions

$$\alpha_i [y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i] = 0, \quad \mu_i \xi_i = 0, \quad (2.19)$$

mean that $\alpha_i > 0$ only for the points on or inside the margin. These are the *support vectors*, and the solution depends on them alone.

The key feature of (2.18) is that the training data enter *only* through the inner products $\mathbf{x}_i^\top \mathbf{x}_j$. Replacing each such inner product with a kernel evaluation $k(\mathbf{x}_i, \mathbf{x}_j) = K_{ij}$ gives the kernelised SVM,

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad (2.20)$$

with decision function

$$f(\mathbf{x}) = \text{sign} \left(\sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right). \quad (2.21)$$

Both the training problem (2.20) and the prediction rule (2.21) reach the data only through kernel values. So the SVM never needs explicit feature vectors: a precomputed kernel matrix $\mathbf{K}$ is all it requires. This is the exact property our method relies on, since it lets us replace $\mathbf{K}$ with the similarity matrix that attention induces.

Multiclass classification. The formulation above is binary, whereas our tasks have several classes. Multiclass predictions come from combining binary SVMs, most commonly in a one-versus-one scheme where a classifier is trained for each pair of classes and the final label is decided by majority vote. Each binary problem is solved through the same kernel matrix, so the multiclass extension keeps the

kernel-only property unchanged.

2.5 Related Work

Our method sits where three lines of research meet: the study of what attention represents, the long-standing link between kernels and neural networks, and the literature on zero-shot and transfer learning. We take each in turn and mark where our approach departs from it.

2.5.1 Attention and what it represents

Attention was introduced as a way of weighing the elements of an input against one another, and it has since become the central component of the Transformer [1] and, in vision, of the Vision Transformer [11]. In these architectures the attention matrix is an internal quantity, computed among the tokens or patches of a single input, used to mix their representations, and then passed on. A separate line of work asks what the attention weights actually encode, debating whether they can be read as explanations of a model’s behaviour; the exchange between “Attention is not Explanation” [12] and “Attention is not not Explanation” [13] is representative, and it establishes that the information in attention weights is worth studying in its own right. We share that starting point but differ in two concrete ways. First, we compute attention *between samples* rather than within a single input, so the matrix describes the geometry of a dataset instead of the internal structure of one example. Second, rather than interpreting the weights, we use them: the output of the attention layer becomes a kernel that is handed to a classifier.

2.5.2 Kernels and neural networks

The link between kernels and neural networks has been explored from several directions. The neural tangent kernel [14] shows that, in a certain infinite-width limit, training a network by gradient descent behaves like a kernel method with a fixed kernel set by the architecture. Deep kernel learning [15] goes the other way, placing a Gaussian-process kernel on top of a neural feature extractor and training the two together, while convolutional kernel networks [16] build kernels whose feature maps mirror the structure of a convolutional network. What these share is that the kernel is either derived from the training dynamics or learned as a dedicated object. Our construction is different, and in a sense more direct:

a trained inter-sample attention layer produces representations, and we build a Gram kernel from them, with a frozen backbone making sure the similarity we exploit comes from attention rather than from feature learning. The grounding for treating the resulting matrix as a kernel is the classical framework of kernel methods [8, 9] reviewed in Section 2.3.

2.5.3 Zero-shot and transfer learning

Attribute-based zero-shot learning classifies unseen categories by describing them through a shared vocabulary of semantic attributes [17], and the AwA2 benchmark we use was introduced precisely to evaluate such methods [5]. A related family maps images and class names into a common semantic space, as in the deep visual-semantic embedding of DeVISE [18], so that an unseen class can be recognised from the embedding of its name. These approaches share a reliance on external side-information, attribute vectors or language embeddings, to bridge seen and unseen classes; an early version of our own work followed this route with a sentence-transformer projection before we discarded it for making the method inconsistent (Section 3.2). The transfer setting we adopt instead reaches zero-shot without any external semantics: the encoder is trained with the target task left out, so the task is genuinely unseen, and a shared attention geometry carries the knowledge across. Using a frozen pretrained network as a general-purpose feature extractor is itself a standard transfer-learning strategy [19]. Finally, our few-shot evaluation is related to metric-based few-shot learning such as prototypical networks [20], which classify by proximity in a learned embedding space; where those methods pair a learned metric with a nearest-prototype rule, we pair a learned similarity with a maximum-margin SVM, keeping one kernel mechanism across all three settings.

2.6 Summary

We now have every piece the method needs. A frozen backbone (Section 2.1) turns images into fixed feature vectors, so that later performance reflects the layers we add. Sample-level attention with tied queries and keys (Section 2.2) produces a similarity structure over those samples. Kernel theory (Section 2.3) tells us that a symmetric PSD matrix is a valid kernel, and that the Gram construction $\mathbf{Z}\mathbf{Z}^\top$ gives one automatically, symmetric and PSD regardless of the attention weights it is built from. Finally, the SVM (Section 2.4) consumes such a matrix directly,

without explicit features. The next chapter puts these pieces together into the four-stage pipeline that defines *attention as a kernel*.

Chapter 3

Proposed Method

This chapter presents the method, which we call *attention as a kernel*. We start with an overview of the pipeline (Section 3.1) and then explain the reasoning behind it by retracing the path that produced it (Section 3.2); each alternative we discarded pins down a requirement the final design meets, so the path is part of the justification rather than a detour. We then describe the four stages in detail (Sections 3.3–3.6), give the full procedure as an algorithm (Section 3.7), and finally explain how the transfer setting realises zero-shot and why a kernel on its own cannot go further (Section 3.8).

3.1 Overview

The method is a four-stage pipeline that turns a collection of images into a kernel and classifies with it. Let N be the number of samples under consideration. The stages are:

1. a *frozen backbone* maps each image to a fixed feature vector;
2. a *full attention layer* with tied queries and keys computes an attention matrix *between* samples and produces an attention output $\mathbf{Z}$;
3. a *kernel* is formed from the row-normalised output as $\mathbf{K} = \hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top$;
4. the precomputed kernel is handed to a *multi-class SVM*.

Figure 3.1 shows the pipeline. Two properties, built in on purpose, do the real work. First, the backbone never trains, so the only learned component is the attention layer, and any performance the kernel reaches can be credited to attention rather than to feature adaptation. Second, the kernel is built from the

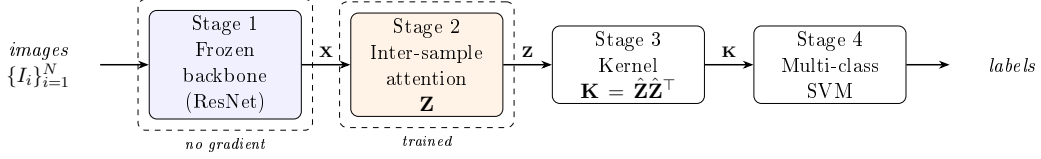


Figure 3.1: The four-stage pipeline. The backbone (blue) is frozen; only the attention layer (orange) is trained. The kernel is built from the attention output $\mathbf{Z}$, whose inter-sample content comes from the attention matrix $\mathbf{A}$, so attention lies at the core of the similarity the SVM uses.

attention output, whose only inter-sample content is produced by the attention matrix $\mathbf{A}$; take attention away and nothing is left for the kernel to relate. The similarity the SVM consumes is exactly the geometry attention has induced over the samples.

3.2 Design rationale: the path to the final method

The final design came out of several iterations, each of which dropped an approach for a concrete reason. We record them because each failure pins down a requirement the final method has to meet.

3.2.1 Kernel from embedding inner products

The first version built the kernel from the inner product of the network’s output embeddings, $\mathbf{S} = \mathbf{Z}\mathbf{Z}^\top$, where $\mathbf{Z}$ was an embedding produced independently of attention. We rejected it: such a kernel is just a linear similarity on the network’s output and does not draw on attention at all. The lesson, and the constraint that shaped everything after, is that the kernel has to originate from the attention computation rather than from an independent embedding. As Section 3.5 shows, the final method satisfies this because the $\mathbf{Z}$ it uses is itself the attention output.

3.2.2 Novelty-based zero-shot

For zero-shot we first used a novelty score: any sample not confidently claimed by a seen task was assumed to belong to the unseen task. This works on single-label datasets, where each sample belongs to exactly one class, and it did so on Cora. It failed on genuinely multi-task datasets, where each sample carries several independent labels at once. When the tasks are independent, removing one gives no signal about it through the others, so a novelty detector has nothing

to detect. This failure is exactly why the datasets were changed to CelebA and AwA2, which are genuinely multi-task (Chapter 4).

3.2.3 Semantic projection for zero-shot

To recover zero-shot on multi-task data, we added a semantic component: the name of each attribute was embedded with a sentence transformer [21], and a small projection was learned from image space into this semantic space. It produced non-trivial results, but it introduced an external language model and made the method inconsistent, since the attention kernel was used for multi-task and few-shot while a different mechanism was used for zero-shot. That inconsistency is why the semantic projection was ultimately removed.

3.2.4 Resolution: transfer as zero-shot

The transfer (leave-one-task-out) setting resolves the tension. Because the encoder is trained without the target task and is frozen with respect to it, the target task is genuinely unseen, so transfer plays the role of zero-shot while using the very same mechanism, the attention kernel, as the other settings. With this change the method becomes uniform across all three evaluation settings, which is the property the first three attempts could not provide.

3.3 Stage 1: Frozen backbone

Each image I_i passes through a backbone f pretrained on ImageNet, whose weights are frozen so no gradient flows through it. The backbone maps the image to a fixed feature vector

$$\mathbf{x}_i = f(I_i) \in \mathbb{R}^{d_{\text{in}}}, \quad \mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d_{\text{in}}}. \quad (3.1)$$

Freezing is a deliberate choice, not a convenience. With f fixed, the feature representation is identical before and after any downstream training, so any performance we later see can be credited to the attention layer and not to the backbone adapting to the task. This keeps the experiment clean and the central claim defensible. In our experiments a ResNet-18 backbone is used for CelebA, while for AwA2 we use the official pre-extracted ResNet-101 features.

3.4 Stage 2: Full inter-sample attention

The second stage is a complete attention operation, with one essential difference from the usual setting: attention is computed *between samples* rather than within a single input. Where ordinary self-attention relates the patches or tokens of one image, here the rows of $\mathbf{X}$ in (3.1) are the feature vectors of *different* images, so the attention matrix expresses relationships across a collection of samples.

Before attention is applied, each feature vector passes through an input projection to the working dimension d of the attention layer,

$$\mathbf{h}_i = \text{Dropout}\left(\text{GELU}(\text{LayerNorm}(\mathbf{W}_{\text{in}}\mathbf{x}_i))\right), \quad (3.2)$$

and the projected rows are collected into $\mathbf{H} \in \mathbb{R}^{N \times d}$. Keys and values are linear projections of $\mathbf{H}$, and the query is tied to the key:

$$\mathbf{K} = \mathbf{H}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{H}\mathbf{W}_V, \quad \mathbf{Q} = \mathbf{K}. \quad (3.3)$$

Attention uses h heads. Splitting the keys into per-head blocks $\mathbf{K}^{(t)} \in \mathbb{R}^{N \times d_{\text{head}}}$ with $d_{\text{head}} = d/h$, the per-head weights are averaged into a single attention matrix,

$$\mathbf{A} = \frac{1}{h} \sum_{t=1}^h \text{softmax}\left(\frac{\mathbf{K}^{(t)}\mathbf{K}^{(t)\top}}{\sqrt{d_{\text{head}}}}\right) \in \mathbb{R}^{N \times N}. \quad (3.4)$$

The stage outputs a residual attention sublayer with an output projection $\mathbf{W}_O$ and layer normalisation:

$$\mathbf{Z} = \text{LayerNorm}(\mathbf{H} + \mathbf{W}_O(\mathbf{A}\mathbf{V})). \quad (3.5)$$

Two choices here decide what follows. Tying the query to the key makes each per-head score matrix $\mathbf{K}^{(t)}\mathbf{K}^{(t)\top}$ symmetric, and averaging over heads keeps it so, so the raw compatibility between two samples does not depend on which one is read as the query. The row-wise softmax in (3.4) then normalises each row by its own sum and does not preserve this symmetry, so $\mathbf{A}$ is in general not symmetric. This does not affect the kernel: what Stage 3 builds is a Gram matrix from the attention output, symmetric and positive semidefinite by construction whatever $\mathbf{A}$ looks like, so the tying is a modelling choice rather than a requirement for a valid kernel. Scaling by $\sqrt{d_{\text{head}}}$ stops the dot products from growing with the dimension and saturating the softmax. The one part of (3.5) in which the attention matrix appears is the term $\mathbf{W}_O(\mathbf{A}\mathbf{V})$; remove it and $\mathbf{Z}$ collapses

to $\text{LayerNorm}(\mathbf{H})$, a per-sample transformation with no relationships between samples at all. That term is what carries information across the collection, and it is why the attention output is the object we build the kernel from.

The attention layer is trained together with one linear classification head per task, under a cross-entropy loss summed over the tasks; this is where the projections $\mathbf{W}_{\text{in}}, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O$ are learned. The heads only drive this training and are dropped at evaluation, where classification is left entirely to the kernel and the SVM.

3.5 Stage 3: Kernel construction

The kernel is built from the attention output $\mathbf{Z}$, which is first ℓ_2 -normalised row-wise and then multiplied by its own transpose:

$$\hat{\mathbf{z}}_i = \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2}, \quad \mathbf{K} = \hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top, \quad K_{ij} = \hat{\mathbf{z}}_i^\top \hat{\mathbf{z}}_j. \quad (3.6)$$

Row normalisation turns each entry into a cosine similarity, so the diagonal is one and every entry lies in $[-1, 1]$; this puts all samples on a common scale and stops a few large-norm rows from dominating the kernel. By the Gram construction of (2.13), $\mathbf{K} = \hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top$ is symmetric and positive semidefinite by construction, and so is a valid kernel in the sense of Section 2.3.

This construction meets the constraint identified in Section 3.2. Since the attention output carries the inter-sample term $\mathbf{W}_O(\mathbf{A}\mathbf{V})$ of (3.5), the kernel depends on the attention matrix directly: strip the attention away and $\mathbf{Z}$ reduces to a per-sample map of the features, leaving the kernel with no cross-sample structure to separate. This is what distinguishes (3.6) from the rejected embedding kernel $\mathbf{Z}\mathbf{Z}^\top$ of Section 3.2: there $\mathbf{Z}$ was an embedding formed independently of attention, whereas here it is the attention output itself.

3.6 Stage 4: Classification with an SVM

The kernel $\mathbf{K}$ is passed to a multi-class SVM that accepts a precomputed kernel, so the classifier touches the data only through $\mathbf{K}$, exactly as in Section 2.4. Multi-class predictions use a one-versus-one scheme over the classes of each task.

The kernel is computed jointly over the union of the training and test samples and then sliced into blocks: the training–training block $\mathbf{K}_{\text{tr},\text{tr}}$ fits the SVM, and the test–training block $\mathbf{K}_{\text{te},\text{tr}}$ is used to predict. Only the test *features* enter the

Algorithm 1 Attention-kernel pipeline

Require: images $\{I_i\}_{i=1}^N$, frozen backbone f , task labels for the seen tasks

- 1: $\mathbf{x}_i \leftarrow f(I_i)$ for all i ▷ Stage 1: fixed features, no gradient
- 2: $\mathbf{H} \leftarrow \text{input-projection}(\mathbf{X})$; $\mathbf{K} \leftarrow \mathbf{H}\mathbf{W}_K$, $\mathbf{V} \leftarrow \mathbf{H}\mathbf{W}_V$, $\mathbf{Q} \leftarrow \mathbf{K}$
- 3: $\mathbf{A} \leftarrow \text{mean-heads softmax}(\mathbf{K}\mathbf{K}^\top / \sqrt{d_{\text{head}}})$; $\mathbf{Z} \leftarrow \text{LayerNorm}(\mathbf{H} + \mathbf{W}_O(\mathbf{A}\mathbf{V}))$
▷ Stage 2
- 4: train the attention layer and multi-class heads on the seen tasks (cross-entropy)
- 5: $\hat{\mathbf{Z}} \leftarrow \text{row-normalise}(\mathbf{Z})$; $\mathbf{K} \leftarrow \hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top$ ▷ Stage 3
- 6: fit a multi-class SVM on $\mathbf{K}$ (precomputed) ▷ Stage 4

kernel; the test *labels* are never used for training. Building the similarity structure over all available samples while withholding all test labels is a *transductive* evaluation, not a data leak: the SVM sees how test samples relate to training samples geometrically, but is never told their classes.

3.7 The complete algorithm

Algorithm 1 summarises the full procedure, from frozen features to a trained kernel classifier.

3.8 Transfer as zero-shot, and a structural limit

The same pipeline supports all three evaluation settings without modification; only the supervision changes. In the multi-task setting the encoder is trained on all tasks and one kernel classifies each of them. In the few-shot setting the encoder is frozen and the SVM is trained with only a handful of labels per class. In the transfer setting the encoder is trained with the target task left out; because the encoder never sees that task, the transfer setting doubles as our zero-shot evaluation.

Why is zero-shot realised through transfer rather than through a label-free procedure? The reason is structural, not a matter of tuning. A kernel is a similarity matrix: it says which samples resemble each other, but not which group of similar samples corresponds to which named class. Turning the geometry into a labelled prediction therefore needs at least one *anchor*, either a few labels or external information such as the meaning of a class name. True label-free zero-shot, where no target label of any kind is available, supplies neither, so the kernel on its own cannot assign its clusters to named classes. The transfer setting supplies exactly the minimal anchor needed: the encoder is trained without the

target task, keeping it genuinely unseen, while a small number of target labels let the SVM attach the geometry to the correct classes. This is why zero-shot is obtained through transfer, and Chapter 4 reports the empirical evidence that the limitation is structural.

Chapter 4

Experiments and Results

This chapter evaluates the method of Chapter 3. We describe the datasets and how their labels become multi-class tasks (Section 4.1), state the evaluation protocol shared by all three settings (Section 4.2), list the implementation details (Section 4.3), and report the results (Section 4.4). We then look, in principle and empirically, at why the attention kernel cannot do label-free zero-shot (Section 4.5), note the limitations (Section 4.6), and summarise (Section 4.7).

4.1 Datasets and Tasks

The method is evaluated on two datasets that are *genuinely multi-task*, meaning each sample carries several independent labels at once. This property is essential: as Section 3.2 explained, it is exactly the setting where the discarded novelty-based approach fails and where the final method has to prove itself.

CelebA. CelebA [4] is a set of face images annotated with 40 binary attributes. We group related attributes into multi-class tasks, so each task is a genuine multi-way classification rather than a binary one. Two tasks are used: *hair colour* (black, blond, brown, grey) and *hair style* (straight, wavy, bald). A sample counts as valid for a task if exactly one class of the group is active, which removes ambiguous or contradictory annotations. Features are extracted with a frozen ResNet-18 backbone.

AwA2. Animals with Attributes 2 (AwA2) [5] contains animal images described by 85 semantic attributes. Following common practice, we use the official pre-extracted ResNet-101 features rather than re-extracting them. Three tasks are used: *habitat*, *diet*, and *movement*. The AwA2 attributes overlap (an animal can

Table 4.1: Multi-class tasks defined on each dataset.

Dataset	Task	Classes
CelebA	Hair colour	black, blond, brown, grey
	Hair style	straight, wavy, bald
AwA2	Habitat	ocean, coastal, desert, forest, plains, arctic
	Diet	meat, fish, vegetation, insects, plankton
	Movement	swims, flies, walks, hops, tunnels

be tagged *ocean*, *coastal*, and *water* at once), so requiring exactly one active class, as in CelebA, would throw away most of the data. Instead the class of a group is assigned by the arg max of the continuous attribute strengths, and a sample is valid for a task if at least one attribute of the group is active. Table 4.1 lists the tasks of both datasets.

Datasets that were set aside. Two datasets tried early on were dropped for concrete reasons. Cora was dropped because it is single-label and therefore not genuinely multi-task: each sample belongs to exactly one class, the setting the method is explicitly not designed for. CUB-200 [22] was dropped because its attributes are too fine-grained and local (distinctions of beak shape or wing pattern) for an inter-sample attention that works on global image features; the similarity structure it produced reflected coarse visual resemblance rather than the localised cues these attributes need, and accuracy stayed near chance. These negative results guided the choice of CelebA and AwA2, whose tasks depend on global visual cues.

4.2 Evaluation Protocol

The same method, an attention kernel followed by an SVM, is used in all three settings; only the supervision given to the encoder and to the SVM changes.

Multi-task. The encoder is trained on all tasks; a single kernel is then extracted and used to classify each task in turn. This setting asks whether one kernel, built once, can serve every task at the same time.

Transfer (leave-one-task-out). The encoder is trained with the target task left out, and its kernel is then applied to that unseen task. Because the target task is never seen during encoder training, this setting also serves as our zero-shot

evaluation, as argued in Section 3.8.

Few-shot. The encoder is frozen and the SVM is trained with only five labelled examples per class.

Performance is reported as *balanced accuracy*, the mean of the per-class recall, which suits class-imbalanced tasks; it is averaged over three random seeds. As described in Section 3.6, the kernel is computed over the union of training and test samples and sliced into blocks, and only test features (never test labels) enter it, so the evaluation is transductive rather than leaking supervision. The SVM is trained with a precomputed kernel using a standard implementation [23].

4.3 Implementation details

Both datasets share the same encoder and the same training recipe; only the backbone and the set of tasks differ. The settings are collected in Table 4.2.

Features. For CelebA each image is resized to 64×64 , normalised with the ImageNet statistics, and passed through a frozen ResNet-18 with its classification head removed, giving a 512-dimensional feature vector. For AWA2 we use the official pre-extracted ResNet-101 features (2048-dimensional), so no images are processed on our side; each image inherits the attribute annotation of its animal class. From each dataset we draw a subset of 8,000 samples and split it 75/25 into 6,000 training and 2,000 test samples.

Encoder. The attention layer works in dimension $d = 128$. Features first pass through the input projection of (3.2); keys, queries and values are then formed as in (3.3), with $h = 4$ heads and therefore $d_{\text{head}} = 32$. The residual output (3.5) feeds one linear head per task. All linear layers use Xavier initialisation, and a dropout of 0.3 is applied in the input projection.

Training. The encoder and the heads are trained with AdamW (learning rate 3×10^{-4} , weight decay 10^{-4}) for 40 epochs, with a batch size of 512 and a cosine-annealing schedule. Each batch sums the cross-entropy losses of the active tasks. In the transfer setting the target task is simply left out of this sum, so the encoder never receives a gradient from it.

Classifier. The kernel of (3.6) is handed to an SVM with a precomputed kernel [23], with penalty $C = 3$, balanced class weights, and the usual one-versus-one

Table 4.2: Architecture and training settings. The encoder and the training recipe are identical across datasets; only the backbone, the feature dimension, and the tasks change.

Component	Setting
Feature backbone	ResNet-18 (CelebA) / ResNet-101 pre-extracted (AwA2), frozen
Feature dimension	512 (CelebA) / 2048 (AwA2)
Samples (train/test)	6,000 / 2,000
Attention dimension d	128
Heads h (d_{head})	4 (32)
Dropout	0.3
Optimiser	AdamW
Learning rate	3×10^{-4}
Weight decay	10^{-4}
Epochs / batch size	40 / 512
LR schedule	cosine annealing
SVM penalty C	3.0
SVM multi-class	one-versus-one, balanced weights
Seeds	0, 1, 2

Table 4.3: Balanced accuracy (mean over three seeds). On both datasets the multi-task kernel exceeds the trained network’s own classification heads.

Setting	CelebA	AwA2
Trained network (all-shot)	0.677	0.869
Multi-task kernel	0.750	0.925
Transfer (leave-one-task-out)	0.737	0.886
Few-shot (5 per class)	0.648	0.914

multi-class scheme. As explained in Section 3.6, the kernel is computed over the union of training and test samples and sliced into blocks, and test labels never enter it. All results are averaged over three seeds.

4.4 Results

Table 4.3 reports balanced accuracy for both datasets across the four conditions, and Figure 4.1 shows the same numbers. The row labelled “trained network” is the baseline: the accuracy of the network’s own linear classification heads, trained on all tasks. The remaining rows are the attention kernel evaluated in the three settings.

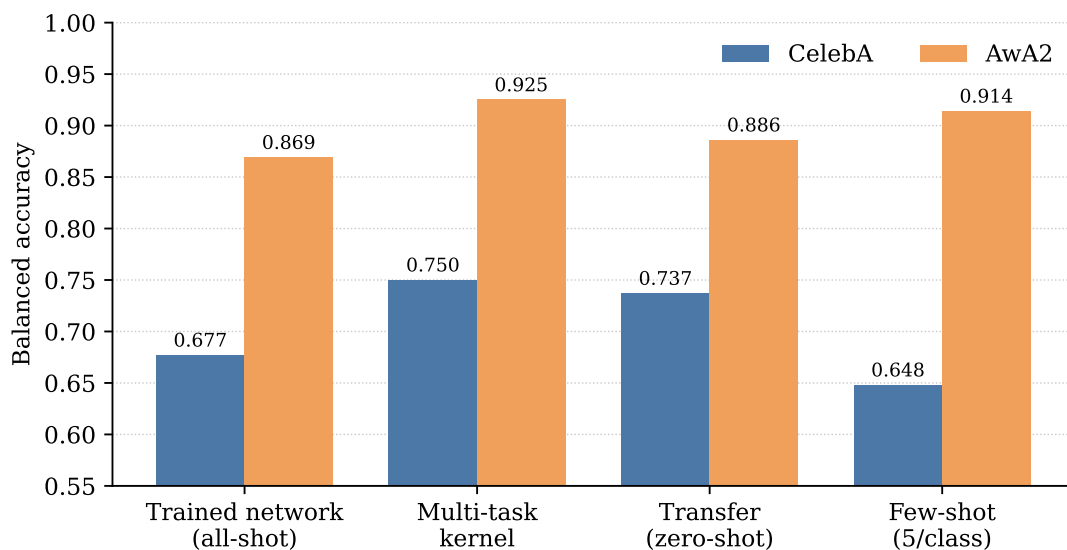


Figure 4.1: Balanced accuracy across the four conditions for CelebA and AwA2. The multi-task kernel (second group) is the highest on both datasets, above the trained network (first group).

The multi-task kernel beats the trained network. On both datasets the multi-task kernel surpasses the trained network’s own classification heads: 0.750 against 0.677 on CelebA, and 0.925 against 0.869 on AwA2. A similarity matrix, read out from inside the model and handed to a classical classifier, thus outperforms the model’s own output layer. The two classifiers use the learned representation differently. The linear heads apply a single hyperplane to the attention output, whereas the SVM draws a maximum-margin boundary over the full similarity structure encoded in the kernel. The geometry the attention layer has induced over the samples thus carries more usable class structure than a linear head can pull from the same representation.

CelebA versus AwA2. The margin by which the kernel beats the baseline is larger and clearer on AwA2. Its tasks, habitat, diet, movement, line up with clear global visual cues, exactly what inter-sample attention over global features captures well, so the induced geometry separates the classes cleanly. CelebA’s tasks depend on more subtle facial attributes, which yields a smaller but still consistent advantage.

Transfer as zero-shot. The transfer results are strong even though the target task is entirely unseen during encoder training: 0.737 on CelebA and 0.886 on AwA2, close to the multi-task numbers and, on AwA2, still above the fully

supervised network baseline. This is the setting that, following the supervisor’s framing, plays the role of zero-shot within one consistent method: the encoder never sees the target task, yet its attention geometry transfers well enough to classify it.

Few-shot. With only five labels per class the kernel stays effective. On AwA2 few-shot accuracy is 0.914, close to the multi-task result and above the network baseline, which suggests the transferred geometry is so well structured that a handful of labels is enough to place the decision boundary. On CelebA it is 0.648, a little below the network baseline, reflecting the subtler class structure of the facial tasks.

Seeing the induced geometry. The claim that attention induces a usable geometry can be read straight off the kernel. Figure 4.2 shows the kernel matrix for one task per dataset, with the samples ordered by class so within-class similarities fall into blocks along the diagonal. On AwA2 the block structure is pronounced: samples of the same habitat have an average similarity of 0.63, against only 0.08 between different habitats, and some habitat pairs are strongly *dissimilar*, down to -0.48 . The kernel has, in effect, already separated the classes before the SVM sees it. On CelebA the same structure is present but much fainter, within-class similarity 0.59 against 0.45 between classes, which is exactly the smaller margin the kernel achieves on the facial tasks. The contrast between the two panels is the same one that appears in the accuracies of Table 4.3: where the induced geometry is crisply block-structured the kernel separates the classes cleanly, and where it is subtler the advantage narrows.

4.5 The limit of label-free zero-shot

For completeness we record a structural limit found during the work, which explains why zero-shot is realised through the transfer setting rather than through a label-free procedure. The argument was sketched in Section 3.8 and is confirmed empirically here.

A kernel is a similarity matrix: it says which samples resemble each other, but not which group of similar samples corresponds to which named class. Turning the geometry into a labelled prediction therefore needs at least one *anchor*, a few labels, or external information such as the meaning of a class name. True label-free zero-shot, where no target label of any kind is available, supplies neither, so

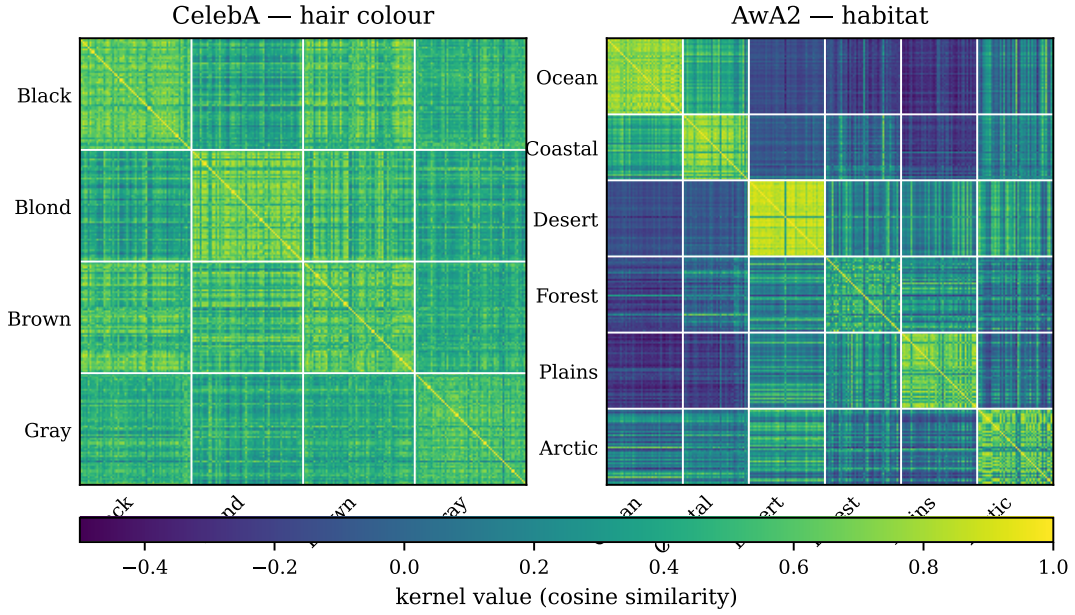


Figure 4.2: Attention kernel matrices for one task per dataset (test samples, ordered by class; brighter means more similar). Bright blocks along the diagonal are groups of same-class samples. The block structure is pronounced on AwA2 (habitat) and much fainter on CelebA (hair colour), mirroring the accuracies in Table 4.3.

Table 4.4: Accuracy of the transferred kernel on an unseen task as a function of the number of anchor labels per class. Performance is near chance when labels are almost absent, confirming that assigning geometry to named classes needs an anchor.

Anchor labels per class	1	3	5
Accuracy	0.37	0.51	0.54

the kernel on its own cannot attach its clusters to named classes.

This is not a matter of tuning. Table 4.4 reports the accuracy of the transferred kernel on an unseen task when only a very small number of labels is available to anchor it. With a single label per class the result is 0.37, below chance; with three and five labels it reaches 0.51 and 0.54, barely above it. The steep dependence on the number of anchor labels, and the near-chance performance when they are almost absent, show that the limit is structural: without an anchor the geometry cannot be assigned to classes at all. This is why zero-shot is obtained through the transfer setting, which supplies the minimal anchor needed, rather than through a label-free method.

4.6 Limitations

Two limitations of the method are worth stating, since both bear on how far the approach can be pushed.

The first is computational. The kernel is a dense $N \times N$ matrix built over the union of training and test samples, so both the memory it needs and the cost of forming it grow quadratically with the number of samples N . Training the SVM on a precomputed kernel is likewise governed by the size of that matrix rather than by a feature dimension. This is why the experiments run on a subset of 8,000 samples per dataset rather than on the full data: the quadratic scaling makes the kernel the binding constraint on problem size long before feature extraction becomes an issue. At the scale considered here this is not a practical obstacle, but it does mean the method, as implemented, does not scale directly to datasets of hundreds of thousands of samples without a way to approximate or sparsify the kernel.

The second is the transductive nature of the evaluation. As described in Section 3.6, the kernel is built over training and test samples jointly, and although test labels are never used, the test *features* do take part in forming the similarity structure. This is a legitimate and common protocol, but it is not the same as a purely inductive setting in which each test point is classified in isolation. A fully inductive variant, where the kernel between a new sample and the training set is computed without the other test samples present, is a natural extension but is not evaluated here.

4.7 Summary

The experiments support the central claim of the thesis. On two genuinely multi-task datasets the attention-induced kernel, fed to an SVM, matches and in fact exceeds the trained network’s own classification heads, and it does so within one method that also handles transfer and few-shot classification. The transfer setting, where the target task is never seen during encoder training, plays the role of zero-shot and gives strong accuracy, while the label-free experiment confirms that a kernel cannot go further without at least a minimal anchor. Overall, the geometry a model induces through attention behaves not as a disposable intermediate quantity but as a transferable structure that carries class information across tasks.

Chapter 5

Conclusion and Future Work

This thesis started from a simple observation: when a model attends to a collection of samples, the attention matrix it produces is already a similarity structure over those samples, and that structure is normally thrown away. We asked whether it could instead be used directly, as a kernel, to drive a classical classifier, and whether the geometry it encodes would carry over to tasks the model was never trained on.

5.1 Summary of contributions

The answer, developed over Chapters 3 and 4, is yes. We introduced *attention as a kernel*: a four-stage pipeline in which a frozen backbone maps each image to a fixed feature vector, a full inter-sample attention layer with tied queries and keys produces an attention output $\mathbf{Z}$, a kernel is built from it as $\mathbf{K} = \hat{\mathbf{Z}}\hat{\mathbf{Z}}^\top$, and a multi-class SVM classifies from that kernel alone. Two properties make the construction meaningful: the backbone never trains, so any performance is attributable to attention rather than to feature adaptation; and the kernel is built from the attention output, whose cross-sample content comes entirely from the attention matrix.

Empirically, on two genuinely multi-task datasets, the attention kernel matched and in fact exceeded the trained network’s own classification heads (CelebA: 0.750 versus 0.677; AwA2: 0.925 versus 0.869). The same method, unchanged, handled three evaluation settings, multi-task, transfer, and few-shot, and the transfer setting, in which the target task is never seen during encoder training, played the role of zero-shot and stayed strong (CelebA: 0.737; AwA2: 0.886). One consistent mechanism therefore covers settings that would otherwise call for separate methods.

A distinctive part of the work was the path to the design. Three earlier attempts were discarded for concrete reasons: a kernel built from independent embeddings, which did not draw on attention at all; a novelty-based zero-shot score, which failed on genuinely multi-task data because independent tasks give no signal about one another; and a semantic projection, which worked but broke the uniformity of the method by bringing in an external language model. Each failure fixed a requirement the final design meets, so recording them is part of the justification for the method rather than a detour.

Finally, we set out a structural limit. Because a kernel expresses which samples resemble each other but not which group corresponds to which named class, it cannot perform label-free zero-shot: assigning geometry to classes needs at least a minimal anchor. The near-chance accuracy we saw when almost no labels are available confirmed that this limit is structural, and it is why zero-shot is realised through the transfer setting rather than through a label-free procedure.

5.2 Future work

Several directions follow naturally from this work.

Beyond a frozen backbone. Freezing the backbone was essential for isolating the contribution of attention, but it also caps the quality of the features the attention layer receives. Allowing controlled fine-tuning of the backbone, or replacing the convolutional backbone with a vision transformer, would show whether the attention kernel keeps beating the corresponding network heads once the underlying representation is itself stronger.

Learning the kernel construction. The kernel is currently formed by a fixed recipe: row normalisation followed by a Gram product. The row-normalisation step, the head-averaging in (3.4), and the choice of how the attention output becomes a similarity are all design decisions that could instead be learned or compared systematically, possibly yielding a better-conditioned kernel.

Richer attribute structure. CUB-200 was set aside because its fine-grained, local attributes are not captured by inter-sample attention over global features. Combining global attention with local or part-based features could extend the method to fine-grained settings, widening the range of tasks for which an attention kernel is useful.

Towards weaker anchors. The structural limit shows that some anchor is unavoidable, but not how weak that anchor may be. Exploring anchors cheaper than labels, a class prototype, a textual description, or a single reference image per class, could push the transfer setting closer to true zero-shot while keeping the method uniform.

Broader validation. The results were obtained on two datasets. Evaluating the attention kernel on further genuinely multi-task benchmarks, and against additional baselines, would strengthen the evidence that the latent geometry attention induces is a transferable structure rather than a property of these particular datasets.

The results point to a single conclusion. The geometry a model induces through attention is not a disposable by-product of computation but a usable, transferable object in its own right, one that a classical kernel classifier can exploit as effectively as, and sometimes more effectively than, the network that produced it.

Bibliography

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [2] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- [4] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738, 2015.
- [5] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.
- [6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [8] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
- [9] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

- [10] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pp. 144–152, 1992.
- [11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference on Learning Representations (ICLR)*, 2021.
- [12] S. Jain and B. C. Wallace, “Attention is not explanation,” in *Proceedings of NAACL*, pp. 3543–3556, 2019.
- [13] S. Wiegrefe and Y. Pinter, “Attention is not not explanation,” in *Proceedings of EMNLP*, pp. 11–20, 2019.
- [14] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [15] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, “Deep kernel learning,” in *AISTATS*, pp. 370–378, 2016.
- [16] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, “Convolutional kernel networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [17] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 951–958, 2009.
- [18] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, “Devise: A deep visual-semantic embedding model,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.
- [19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [20] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

- [21] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of EMNLP*, pp. 3982–3992, 2019.
- [22] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The Caltech-UCSD Birds-200-2011 dataset,” Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.