```
> with(DifferentialGeometry):
  with(Tensor):
```

First we initialize the base manifold.

```
> DGEnvironment[Manifold]([x,y],M);
```

Manifold: M **(1)**

Here is the infinitesimal generator of the symmetry: a vector field on M.

```
M > sym:=evalDG(g1(x,y)*D_x+g2(x,y)*D_y);
```

$$sym := g1(x, y)\ \partial_x + g2(x, y)\ \partial_y \tag{2}$$

An image is modeled as a smooth section of a Fibered Manifold. Initializing the FM:

```
M > DGEnvironment[FiberedManifold]([x,y],[R,G,B],Color);
```

Fibered Manifold: Color **(3)**

We will need to differentiate smooth sections, so here is a (zero) connection:

```
Color > Gamma3:=eval(Connection([nabla(D_x,D_R)=evalDG(a*D_R)
        ]),a=0);
```

$$\Gamma 3 := \nabla_{\partial_x} \partial_R = 0\ \partial_R \tag{4}$$

An image is a smooth section of the fiber bundle.

```
Color > image:=evalDG(f1(x,y)*D_R + f2(x,y)*D_G + f3(x,y)*
        D_B);
```

$$image := f1(x, y)\ \partial_R + f2(x, y)\ \partial_G + f3(x, y)\ \partial_B \tag{5}$$

Clumsy code: we need Maple to recognize that the vector field lives in the base space.

```
Color > symC:=subs(M=Color,sym);
```

$$symC := g1(x, y)\ \partial_x + g2(x, y)\ \partial_y \tag{6}$$

The classification model can be seen as a function of the image with 10 outputs (10 classes).

```
Color > DGEnvironment[FiberedManifold]([R,G,B],[seq(E||i,i=1.
        .10)],EN);
```

Fibered Manifold: EN **(7)**

As before, we define a (zero) connection for later differentiation.

```
EN > GammaN:=eval(Connection([nabla(D_R,D_E1)=evalDG(a*D_E1)]),
     a=0);
```

$$GammaN := \nabla_{\partial_R} \partial_{E1} = 0\, \partial_{E1} \tag{8}$$

Here are the coefficients of the model: 10 functions of an image.

```
EN > funcs:=[seq(1/(1+exp(-F||i(R,G,B))),i=1..10)];
```

$$funcs := \left[\frac{1}{1+e^{-F1(R,G,B)}}, \frac{1}{1+e^{-F2(R,G,B)}}, \frac{1}{1+e^{-F3(R,G,B)}}, \frac{1}{1+e^{-F4(R,G,B)}}, \frac{1}{1+e^{-F5(R,G,B)}}, \frac{1}{1+e^{-F6(R,G,B)}}, \frac{1}{1+e^{-F7(R,G,B)}}, \frac{1}{1+e^{-F8(R,G,B)}}, \frac{1}{1+e^{-F9(R,G,B)}}, \frac{1}{1+e^{-F10(R,G,B)}} \right] \tag{9}$$

Below is the derivative of the model with respect to the image.

```
EN > grad:=CovariantDerivative(DGzip(funcs,DGinformation(EN,
     "FrameFiberVectors"),"plus"),GammaN):
```

Preparing to realize this as a function of x and y:

```
EN > eqns:=[R=f1(x,y),G=f2(x,y),B=f3(x,y)];
```

$$eqns := [R = f1(x,y),\ G = f2(x,y),\ B = f3(x,y)] \tag{10}$$

Now realizing it as a function of x and y.

```
EN > Grad:=DEtools:-dsubs(eqns,grad):
```

The chain rule applies: we have to multiply by the derivative of the image w.r.t. x and y.

```
EN > dd:=DirectionalCovariantDerivative(symC,image,Gamma3);
```

$$dd := \left(g1(x,y) \left(\frac{\partial}{\partial x} f1(x,y)\right) + g2(x,y) \left(\frac{\partial}{\partial y} f1(x,y)\right)\right) \partial_R + \left(g1(x,y) \left(\frac{\partial}{\partial x} f2(x,y)\right) + g2(x,y) \left(\frac{\partial}{\partial y} f2(x,y)\right)\right) \partial_G + \left(g1(x,y) \left(\frac{\partial}{\partial x} f3(x,y)\right) + g2(x,y) \left(\frac{\partial}{\partial y} f3(x,y)\right)\right) \partial_B \tag{11}$$

Another clumsy coding moment: converting this to the base space of the EN vector bundle.

```
Color > DD:=DGzip(DGinformation(dd,"CoefficientList","all"),
        DGinformation(EN,"FrameBaseVectors"),"plus");
```

$$DD := \left(g1(x,y)\left(\frac{\partial}{\partial x} f1(x,y)\right) + g2(x,y)\left(\frac{\partial}{\partial y} f1(x,y)\right)\right)\partial_R + \left(g1(x,y)\left(\frac{\partial}{\partial x} f2(x,y)\right) + g2(x,y)\left(\frac{\partial}{\partial y} f2(x,y)\right)\right)\partial_G + \left(g1(x,y)\left(\frac{\partial}{\partial x} f3(x,y)\right) + g2(x,y)\left(\frac{\partial}{\partial y} f3(x,y)\right)\right)\partial_B \quad (12)$$

Now we "multiply" the terms together.

```
EN > ans2:=ContractIndices(Grad,DD,[[2,1]]):
```

Is this the right answer? let's check. The model is a smooth section of pi:R10 -> R2.

```
EN > DGEnvironment['VectorSpace'](10,V10);
```

Vector Space: V10 **(13)**

```
V10 > DGEnvironment[VectorBundle](M,V10,N);
```

Vector Bundle: N **(14)**

Once again, we will need to differentiate, so we define a (zero) connection.

```
N > Gamma:=eval(Connection([nabla(D_x,E1)=a*E1]),a=0);
```

$$\Gamma := \nabla_{\partial_x} E1 = 0\, E1 \quad (15)$$

The functions given before are, at the core, functions of x and y:

```
N > Funcs:=subs(eqns,funcs);
```

$$Funcs := \left[\frac{1}{1+e^{-F1(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F2(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F3(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F4(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F5(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F6(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F7(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F8(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F9(f1(x,y),\, f2(x,y),\, f3(x,y))}}, \frac{1}{1+e^{-F10(f1(x,y),\, f2(x,y),\, f3(x,y))}}\right] \quad (16)$$

Here is the neural network. The functions F are learned during training.

```
N > Model:=DGzip(Funcs,DGinformation(N,"FrameFiberVectors"),
    "plus");
```

$$Model := \frac{1}{1+e^{-F1(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E1 + \frac{1}{1+e^{-F2(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E2 + \frac{1}{1+e^{-F3(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E3 + \frac{1}{1+e^{-F4(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E4 + \frac{1}{1+e^{-F5(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E5 + \frac{1}{1+e^{-F6(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E6 + \frac{1}{1+e^{-F7(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E7 + \frac{1}{1+e^{-F8(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E8 + \frac{1}{1+e^{-F9(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E9 + \frac{1}{1+e^{-F10(f1(x,y),\, f2(x,y),\, f3(x,y))}}\, E10 \tag{17}$$

Awkward coding again: need to recognize the vector field as living on the base space of N.

```
N > symCn:=subs(Color=N,symC);
```

$$symCn := g1(x, y)\, \partial_x + g2(x, y)\, \partial_y \tag{18}$$

This is the real answer:

```
N > ans:=DirectionalCovariantDerivative(symCn,Model,Gamma):
```

Is it equal to what we got using the product rule? We can check, but there's one last awkward coding moment: realizing our previous answer as a section of the bundle N (which is equivalent to EN).

```
N > ans3:=convert(DGzip(DGinformation(ans2,"CoefficientList",
    "all"),DGinformation(N,"FrameFiberVectors"),"plus"),DGtensor)
    :
N > simplify(evalDG(ans-ans3));
```

$$0 \tag{19}$$

The quantities are equal, meaning the directional covariant derivative of the model along a vector field can be computed using the product rule.