```
classdef subtractionChannelLayer < nnet.layer.Layer & nnet.layer.Formattable & nnet.↙
layer.Acceleratable % (Optional)
    %SUBTRACTIONCHANNELLAYER Custom layer to be used in Sensor Fusion net
    %
    %   Authors:
    %       Michael Caiola (Michael.Caiola@fda.hhs.gov)
    %       Meijun Ye (Meijun.Ye@fda.hhs.gov)
    %
    %   Disclaimer: This software and documentation (the "Software") were
    %   developed at the Food and Drug Administration (FDA) by employees of
    %   the Federal Government in the course of their official duties.
    %   Pursuant to Title 17, Section 105 of the United States Code,
    %   this work is not subject to copyright protection and is in the
    %   public domain. Permission is hereby granted, free of charge, to any
    %   person obtaining a copy of the Software, to deal in the Software
    %   without restriction, including without limitation the rights to
    %   use, copy, modify, merge, publish, distribute, sublicense, or sell
    %   copies of the Software or derivatives, and to permit persons to
    %   whom the Software is furnished to do so. FDA assumes no
    %   responsibility whatsoever for use by other parties of the Software,
    %   its source code, documentation or compiled executables, and makes
    %   no guarantees, expressed or implied, about its quality,
    %   reliability, or any other characteristic. Further, use of this code
    %   in no way implies endorsement by the FDA or confers any advantage
    %   in regulatory decisions. Although this software can be
    %   redistributed and/or modified freely, we ask that any derivative
    %   works bear some notice that they are derived from it, and any
    %   modified versions bear some notice that they have been modified.

    properties
        % (Optional) Layer properties.
    end

    properties (Learnable)
        % (Optional) Layer learnable parameters.

        % Declare learnable parameters here.
    end

    properties (State)
        % (Optional) Layer state parameters.

        % Declare state parameters here.
    end

    properties (Learnable, State)
        % (Optional) Nested dlnetwork objects with both learnable
        % parameters and state parameters.

        % Declare nested networks with learnable and state parameters here.
    end
```

```
    methods
        function layer = subtractionChannelLayer(numInputs,args)
            % (Optional) Create a myLayer.
            % This function must have the same name as the class.

            arguments
                numInputs
                args.name = '';
            end
            name = args.name;
            layer.NumInputs = numInputs;
            layer.Name = name;
            layer.Description = "Subtract two layers.";
            layer.Type = "Subtraction";

        end


%         function layer = initialize(layer,layout)
%             % (Optional) Initialize layer learnable and state parameters.
%             %
%             % Inputs:
%             %         layer  - Layer to initialize
%             %         layout - Data layout, specified as a networkDataLayout
%             %                  object
%             %
%             % Outputs:
%             %         layer - Initialized layer
%             %
%             %  - For layers with multiple inputs, replace layout with
%             %    layout1,...,layoutN, where N is the number of inputs.
%
%             % Define layer initialization function here.
%         end


        function Z = predict(~,X1,X2)
            % Forward input data through the layer at prediction time and
            % output the result and updated state.
            %
            Z = X1-X2;

        end



%         function layer = resetState(layer)
%             % (Optional) Reset layer state.
%
%             % Define reset state function here.
```

```
%         end
%
%         function [dLdX,dLdW,dLdSin] = backward(layer,X,Z,dLdZ,dLdSout,memory)
%             % (Optional) Backward propagate the derivative of the loss
%             % function through the layer.
%             %
%             % Inputs:
%             %         layer   - Layer to backward propagate through
%             %         X       - Layer input data
%             %         Z       - Layer output data
%             %         dLdZ    - Derivative of loss with respect to layer
%             %                   output
%             %         dLdSout - (Optional) Derivative of loss with respect
%             %                   to state output
%             %         memory  - Memory value from forward function
%             % Outputs:
%             %         dLdX   - Derivative of loss with respect to layer input
%             %         dLdW   - (Optional) Derivative of loss with respect to
%             %                  learnable parameter
%             %         dLdSin - (Optional) Derivative of loss with respect to
%             %                  state input
%             %
%             %  - For layers with state parameters, the backward syntax must
%             %    include both dLdSout and dLdSin, or neither.
%             %  - For layers with multiple inputs, replace X and dLdX with
%             %    X1,...,XN and dLdX1,...,dLdXN, respectively, where N is
%             %    the number of inputs.
%             %  - For layers with multiple outputs, replace Z and dlZ with
%             %    Z1,...,ZM and dLdZ,...,dLdZM, respectively, where M is the
%             %    number of outputs.
%             %  - For layers with multiple learnable parameters, replace
%             %    dLdW with dLdW1,...,dLdWP, where P is the number of
%             %    learnable parameters.
%             %  - For layers with multiple state parameters, replace dLdSin
%             %    and dLdSout with dLdSin1,...,dLdSinK and
%             %    dLdSout1,...,dldSoutK, respectively, where K is the number
%             %    of state parameters.
%             %
%             % Define layer backward function here.
%         end
    end
end
```