b/26c967141387aaff0363ffb524afee5d9034481f/jax/_src/lax/lax.py#L3170

```
3170 def _conv_general_dilated_translation_rule(
3171     c, lhs, rhs, *, window_strides, padding,
3172     lhs_dilation, rhs_dilation, dimension_numbers, feature_group_count,
3173     batch_group_count, precision, expand_complex_convolutions,
3174     preferred_element_type, **unused_kwargs):
3175   assert type(dimension_numbers) is ConvDimensionNumbers
3176   dimension_numbers = _conv_general_proto(dimension_numbers)
3177   precision_config = _precision_config(precision)
3178   dtype = c.get_shape(lhs).numpy_dtype()
3179   if expand_complex_convolutions and np.issubdtype(dtype, np.complexfloating):
3180     # We use a trick for complex multiplication due to Gauss which uses three
3181     # multiplications and five additions; instead of the naive method of four
3182     # multiplications and two additions.
3183     # https://en.wikipedia.org/wiki/Multiplication_algorithm#Complex_multiplication_algorithm
3184     #
3185     # This performance win comes with a trade-off in accuracy; especially in
3186     # cases when the real and imaginary differ hugely in magnitude. The relative
3187     # error bound (e.g. 1p-24 in case of float32) would be relative to the
3188     # maximum of real and imaginary parts of the result instead of being
3189     # satisfied by the real and imaginary parts independently of each other.
3190     if preferred_element_type is not None:
3191       # Convert complex dtype to types used for real and imaginary parts
3192       assert np.issubdtype(preferred_element_type, np.complexfloating)
3193       preferred_element_type = xla_client.dtype_to_etype(
3194           np.float64 if preferred_element_type == np.complex128 else np.float32)
3195
3196     conv = lambda x, y: xops.ConvGeneralDilated(
3197         x, y, window_strides, padding, lhs_dilation, rhs_dilation,
3198         dimension_numbers, feature_group_count, batch_group_count,
3199         precision_config=precision_config,
3200         preferred_element_type=preferred_element_type)
3201     lhs_real, lhs_imag = xops.Real(lhs), xops.Imag(lhs)
3202     rhs_real, rhs_imag = xops.Real(rhs), xops.Imag(rhs)
3203     k1 = conv(xops.Add(lhs_real, lhs_imag), rhs_real)
3204     k2 = conv(lhs_real, xops.Sub(rhs_imag, rhs_real))
3205     k3 = conv(lhs_imag, xops.Add(rhs_real, rhs_imag))
3206     return xops.Complex(xops.Sub(k1, k3), xops.Add(k1, k2))
3207
3208   if preferred_element_type is not None:
3209     preferred_element_type = xla_client.dtype_to_etype(preferred_element_type)
3210
3211   return xops.ConvGeneralDilated(
3212       lhs, rhs, window_strides, padding, lhs_dilation, rhs_dilation,
3213       dimension_numbers, feature_group_count, batch_group_count,
3214       precision_config=precision_config,
3215       preferred_element_type=preferred_element_type)
3216
3217 def _conv_general_dilated_batch_rule(
```